Sonderheft Nr. 204
Preis 28 DM
28 sfr., 210 öS
mc

Die Serie wurde vom Bundesministerium für Forschung und Technologie gefördert

MIKRO-COMPUTER 2

Schritt für Schritt

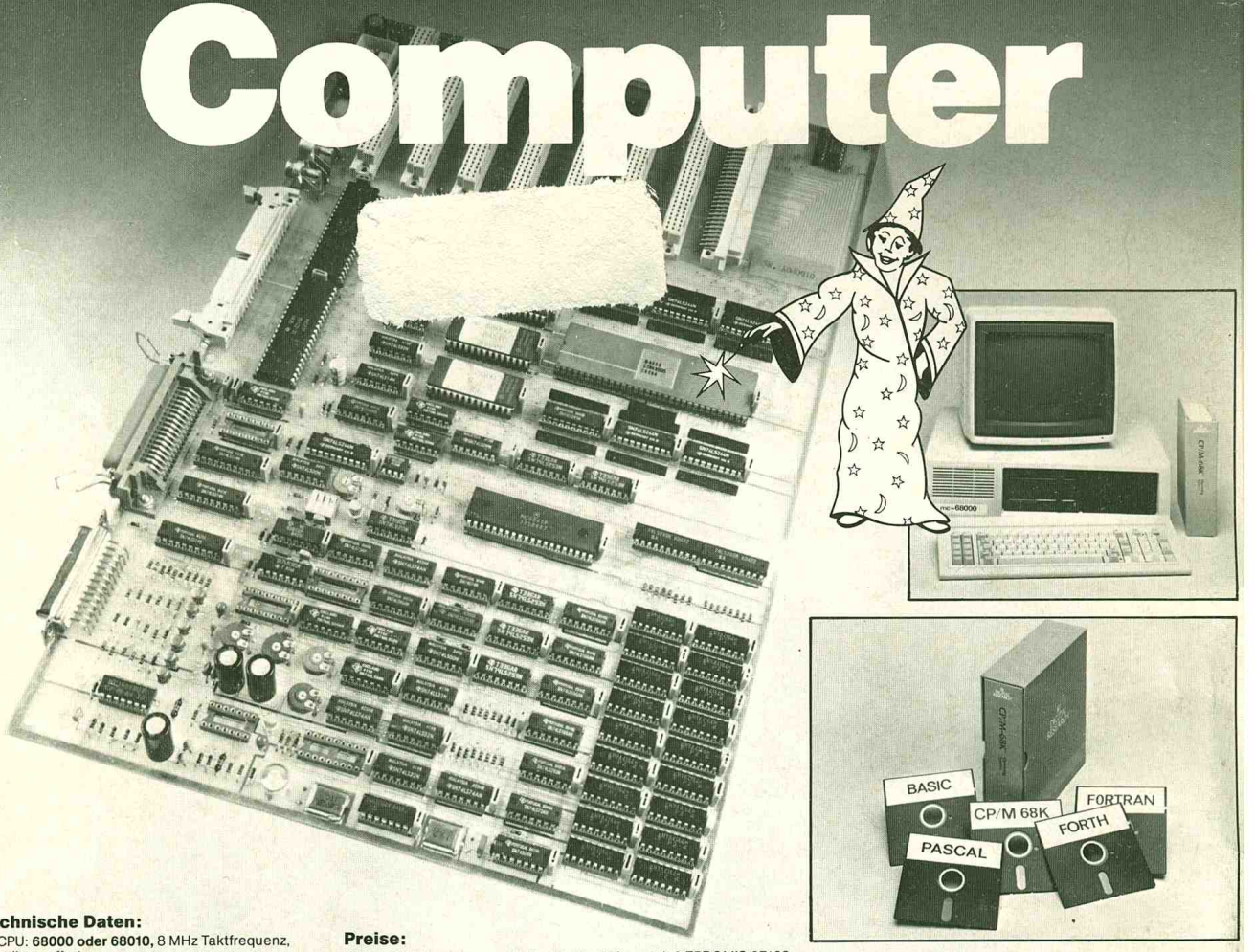
Der Vollausbau des NDR-Computers

Z80 und CP/M: Von der Hardware zum Betriebssystem

68008 und CP/M: Vom Maschinenbefehl bis zum Floppy-Betrieb alles verständlich

...roelektronik"
...weite Sonderheft zur Fernsehserie

Der mc 68000 Computer



Technische Daten:

- ☆ CPU: **68000 oder 68010,** 8 MHz Taktfrequenz, voll gepuffert.
- ☆ RAM: on board 128/512 KB, über Slots **erweiterbar bis auf 16 MByte.**
- ☆ ROM: 64 KByte on board, über Slots beliebig erweiterbar.
- ☆ Flexibles Design durch programmierbaren Adreßdekoder.
- ☆ Systembus auf 8 Slots herausgeführt, VME-Bus ähnlich, jedoch für den mc 68000 optimiert.

Schnittstellen:

- ☆ **V24-Schnittstelle** für seriellen Drucker oder Terminal
- ☆ **Centronics** parallel für Druckeranschluß
- ☆ **Tastaturport** für parallele oder serielle Tastatur
- ☆ **Kassetteninterface mit 3600 Baud**
- ☆ **16 Bit Userport** (mit Handshakeleitungen und Videosignalen zur freien Verdrahtung)
- ☆ Anschluß für **Analog-Joystick** (bzw. 2 Paddles)
- ☆ **Stereo-Tonausgang** (vorbereitet für Sprachausgabe)
- ☆ **Video:** BAS 16 MHz **80 Zeichen x 25 Zeilen,** Zeichenmatrix **8 x 8 oder 8 x 10 Punkte, Vollgrafik mit 640 x 200 Punkten,** Hell/Dunkel Attribut (Bildformate frei programmierbar). Hard/Soft-Scroll. Nutzbare Pixelzahl 1024 x 1024, direkter DMA auf das Video-RAM ermöglicht **superschnelle Grafiken.** Bis zu 32 Grafik-Seiten on Board.
- ☆ **Farbgrafik:** F BAS, RGB: 8 MHz, 40 Zeichen (farbig) x 25 Zeilen bzw. **8 Farben Vollgrafik.**
- ☆ Anschluß für Farb-TV über Modulator bzw. Video-Buchse.)

Software: (on board)

- ☆ **32 KByte Betriebssystem** in zwei EPROM'S enthält: Bildschirm-Editor, Monitor, **symbolischer Assembler mit Debugger** DOS Urlader für CP/M 68K**, I/O-Treiber, Kassetten-Interface (großzügiger Ausbau durch Vektortechnik vorgesehen).

Preise:

● **mc-68K-Board**	Platine (380 x 250 mm) + 2 EPROM'S 27128 **mit Betriebssystem** + Handbuch	DM 498,–
	Handbuch einzeln	DM 30,–
● **mc-68K-Kit**	Komplett-Bausatz (Platine, **Betriebssystem,** alle Bauteile, 128K RAM, Präzisionssockel, Stecker etc. + Handbuch)	DM 1798,–
● **mc-68K-Floppy-Karte**	Komplett-Bausatz mit allen Bauteilen und Platine, Anschluß für 8 Laufwerke (4 x 5¼" u. 4 x 8") alle Formate	DM 398,–
● **mc-68K-System I,**	kompletter 68000 Computer (128 KB RAM) mit IBM/PC-kompatibler Tastatur, 22 MHz-Monitor, Floppy-Disk-Laufwerk (1 MegaByte unformatiert) im Gehäuse **inkl. CP/M68K**** Startbereit:	DM 8500,–
● **mc-68K-Gehäuse,** (siehe Bild)	formschönes Stahlblechgehäuse (500 x 410 x 140 mm)	DM 198,–
● **mc-68K-Monitor,**	Monitor grün oder bernstein (22 MHz)	DM 398,–
● **mc-68K-Tastatur,**	Cherry-Super-Tastatur! IBM-PC kompatibel (siehe Bild) inkl. Interface und Gehäuse	DM 398,–
● **mc-68K-Floppy 5¼"**	**1 MegaByte** unformatiert, DS, 80 Track	DM 698,–
● **mc-68K-Netzteil 1,**	+ 5V/6A; +12V/2A; −12V/1A; −5V/0,5A	DM 248,–
● **mc-68K-Netzteil 2,**	+5V/6A; +12V/4,5A; −12/0,5A; −5V/0,5A	DM 356,–
● **mc-68K-Netzteil 3,**	+5V/7,5A; +12V/5A; +12/1,5A; −12V/0,5A	DM 598,–

Software:

● CP/M 68K für mc-68000 angepaßt (incl. C-Compiler) V 1.2	DM 698,–
● 68K-Emulator für CP/M80 und Z80	DM 398,–
● 68K-Basic	DM 298,–
● 68K-Pascal – S/128K	DM 298,–
● 68K-Fortran 77	a.A.
● 68K-Forth	a.A.

alle Preise incl. 14% MwSt. Zwischenverkauf vorbehalten.

Software: (auf Diskette)

- ☆ CP/M 68 K V 1.2 incl. C-Compiler
- ☆ OS-9/68 K*
- ☆ UNIX*
- ☆ **Emulator für CP/M 80 und Z 80,** (ein exzellentes Hilfsmittel, um Ihre CP/M 80 Software auf einem 68000-System weiter zu verwenden).
- ☆ C-Compiler
- ☆ BASIC 68K
- ☆ BASIC 68K Plus
- ☆ CBASIC 68K
- ☆ PASCAL 68K
- ☆ FORTRAN 68K
- ☆ FORTH

In Vorbereitung:

- ☆ BASIC-Compiler, mit Unterstützung strukturierter Programmiertechniken (im EPROM).
- ☆ RAM-Erweiterungskarte (256K/1 Megabyte).
- ☆ Arithmetikprozessor-Karte
- ☆ Sprachausgabe mit dem SC 02.
- ☆ HIGH-RES.-Grafik-Karte

* = in Vorbereitung
** = CP/M 68K ist ein Warenzeichen von Digital Research

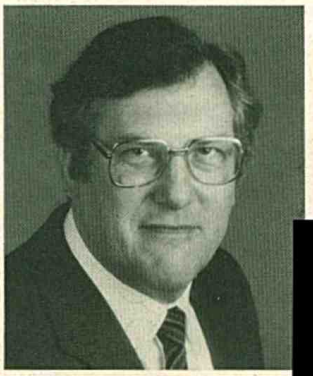
Prospekt anfordern bei:

Heninger Digital Service
Vertriebsgesellschaft für
digitale elektronische Bauteile mbH
Landwehrstr. 37
8000 München 2
Tel. (089) 59 69 16
Telex 5 23 545 hends d

Heninger DigitalService

[redacted]

Vorwort



[redacted]

Dieses zweite Begleitheft zur Fernsehreihe „Mikroelektronik" ist für die vielen Zuschauer und Leser geschrieben, die nach den ersten Schritten anhand unserer Sendungen und des Begleitmaterials Feuer gefangen haben. In diesem zweiten Heft ist alles zusammengefaßt, was ein angehender Professional an Kenntnissen benötigt, um sowohl praktisch als auch von der Theorie her auf dem Gebiet der Mikroelektronik fit zu werden. Sie können den ganzen Stoff des Heftes im wahrsten Sinn des Wortes begreifen, wenn Sie die vorgeschlagenen Baugruppen nachbauen. Das System ist dabei so reichhaltig, daß für jeden Bedarf und jeden Grad der Ausbildung der richtig dimensionierte Mikrocomputer dabei ist. Wir haben dabei darauf geachtet, die Vorschläge einerseits so allgemein zu halten, daß sie als beispielhaft für das gesamte Gebiet Mikroelektronik gelten können. Die Computer, die Sie dabei entwickeln, sind andererseits auf dem Stand der Technik und entsprechend leistungsfähig. Die Modularität des Systems gibt Ihnen die Sicherheit, daß auch in Zukunft Ihr Rechner auf den jeweils neuesten Stand der Technik nachgerüstet werden kann. Weil das Interesse an dem Medienverbundsystem Mikroelektronik (bestehend aus Fernsehen, Begleitheften, Buch und Bausätzen sowie Kursen) so groß ist, können wir kaum die Anfragen und Anregungen aus Ihrem Kreis (und dem der Lehrer) bewältigen. Wir werden uns bemühen, das didaktische Konzept der Sendung und das System der Baugruppen, den gesamten Medienverbund also, immer wieder dem Technischen Fortschritt anzupassen. Sie nehmen teil an einer aufregenden Entwicklung, an der Sie mit eigenen Beiträgen mitwirken können. Sollten Sie neu in die Mikroelektronik eingestiegen sein, so werden Sie feststellen, daß es nur mit viel Arbeit gelingt, in das Gebiet einzudringen. Ein stolzer Computerbesitzer, der sein Gerät durch und durch kennt und beherrscht, das können Sie bei genügend Geduld und Ausdauer werden.

Inhalt

Vorwort 3
Impressum 3
Architektur und Sprache 5
(Mehr Bits für die CPU)
Sprachprobleme 16
Schleifen und Roboter 24
Die Landung auf dem Mond . . . 32
Tick-Tack 41
Ausflug nach Simland 48
Krater in Pascal 57
Die Türme von Hanoi 63
Taschenrechner selbst gebaut . . 68
Die Befehle des 68008 71
Die Befehle des 68008-Grundprogrammes in Kürze . . . 72
Die Z80-CPU voll ausgebaut . . . 74
Die Floppy-Karte 80
Flomon, das Z80-Programm für die Floppy 87
Das BIOS für FLO-2 92
68008: Arbeiten mit der Floppy Disk 104
Der Grafikprozessor EF 9366 . . 123

Impressum

1985

Franzis-Verlag GmbH, Karlstraße 37–41, D-8000 München 2.

Bearbeitet von der Redaktion der Zeitschrift mc.
Für den Text verantwortlich: Dipl.-Math. Ulrich Rohde.

© Sämtliche Rechte – besonders das Übersetzungsrecht – an Text und Bildern vorbehalten. Fotomechanische Vervielfältigung nur mit Genehmigung des Verlages. Jeder Nachdruck, auch auszugsweise, und jede Wiedergabe der Abbildungen, auch in verändertem Zustand, sind verboten.

ISSN 0722-0022.

Druck: Franzis-Druck GmbH, München.
Printed in Germany. Imprimé en Allemagne.

ZV-Art.-Nr. 20404 · F/ZV/285/1089/9'

FRANZIS FSS SOFTWARE SERVICE

Arbeitsmaterialien zur Programmerstellung:

n00 Schaltpläne und Unterlagen 6.--
Das Heft enthält Schaltpläne (auch zum 68008), kurze Versuchsbeschreibungen, Bestückungspläne, Beschreibung der 68008-Programme, Berichtigungen und Datenblätter. (Solange Vorrat reicht !)

n01 Z-80 Grundprogramm 12.--
160 Seiten Assemblerlisting des Grundprogramms

n02 Z-80 Aufbauprogramme 12.--
160 Seiten Programmlistings der Versuche aus der Sendung und die Programme aus den EPROMs.

n03 68008 Grundprogramm 12.--
Die "Programmbibliothek mit dem Assembler und Editor, dem Bibliotheksmodul und den anderen Grundprogramm-Routinen

n04 68008 Aufbauprogramme 12.--
Die Versuche mit dem 68008: Assembler- und Pascalprogramme

n05 68008 Pascal 12.--
Ein kurzer Abriß der Sprache Pascal, die Beschreibung von Pascal-S, einige Beispiele und das Listing des Pascal-S-Systems auf über 200 Seiten.

EPROMs zur Fernsehserie
Wenn Sie die EPROMs nicht schon mit den Bausätzen erhalten haben, bekommen Sie sie beim FSS:

Zur SBC-II-Karte

ndr01 Z-80-Grundprogramm (2 x 2732) 75.--

ndr02 SCOP 29.--
Einstellen der Kassettenschnittstelle

ndr03 Basic (2 x 2732) 75.--
Das BASIC entspricht dem Mikrosoft-Basic mit allen Funktionen. Mit Handbuch!

ndr04 Z-80 GOSI (2 x 2732) 75.--
GOSI entspricht einem LOGO ohne Listenverarbeitung. Durch den deutschen Befehlssatz das Ideale für den Einstieg. Da die E/A komfortabel angesprochen werden kann, ist GOSI auch für Steuerungsaufgaben bestens geeignet.

ndr05 SPS (2 EPROMs 2732) 90.--
Speicherprogrammierte Steuerungen Mit Handbuch. Siehe auch ELO 2 und 3/1985.

Die Kassettenaufzeichnung und Druckerausgabe wird bei allen Programmen unterstützt. Sowohl vom Basic als auch vom GOSI sind auf Anfrage die Programmlistings in gedruckter Form erhältlich.

Zur Vollausbau-CPU-Karte (Z-80) mit ROA64

ndr06 Z-80-Grundprogramm im 2764 75.--
Speicherbereich ($0000-$1FFF)

ndr07 Z-80-Grundprogramm im 2764 75.--
Speicherbereich ($2000-$3FFF)

ndr08 GOSI im 2764 ($2000-$3FFF) 75.--

ndr09 BASIC im 2764 ($4000-$5FFF) 75.--

ndr10 Zeilenassembler im 2764 75.--
Speicherbereich ($6000-$7FFF). Zu ndr05!

ndr11 Zeilenassembler im 2764 75.--
Speicherbereich ($6000-$7FFF). Zu ndr06!

ndr12 FLOMON zum Floppybetrieb 75.--

Sie können so alle Programmsysteme nebeneinander verwenden !

Zur CPU 68008

ndr20 68008 Grundprogramm Vers. 4.3 185.--
In 4 EPROMs 2764 m. Handbuch

ndr21 68008 Pascal 185.--
In 4 EPROMs 2764 m. Handbuch

ndr22 68008 GOSI68K-Compiler 185.--
In 3 EPROMs 2764 m. Handbuch

ndr23 68008 Boot-Prom für CP/M68K 29.--

ndr24 68008 Formatierer UFORM68K 75.--

Betriebssysteme

com22/5 Betriebssystem CP/M 2.2 398.--
Für Z-80 mit deutsche Kurzanleitung auf Diskette 5 1/4"
Bitte geben Sie an, welches Format Sie möchten:
ECMA 70 (40-Spur, einseitig) oder mc-Format (80-Spur, doppelseitig)

com22/8 Betriebssystem CP/M 2.2 398.--
8"-Diskette (STD-IBM-Format)

com68/5 Betriebssystem CP/M68K 698.--
Für 68008/68000 mit Handbuch auf Diskette 5 1/4". Formate wie bei CP/M 2.2!

com68/8 Betriebssystem CP/M68K 698.--
8"-Diskette (STD-IBM-Format)

com22/d Deutsches Handbuch für CP/M 2.2 56.--

com68/a Amerikanisches Handbuch für CP/M68K einzeln 248.--

Sonstiges

Da diejenigen unter Ihnen, die gleich mit dem 68008 anfangen, das SCOP-Eprom nicht verwenden können, haben wir einen **Einstellservice für die Kassettenplatine** eingerichtet.

Die Kassettenplatine muß funktionsfähig sein, mit Diodenstecker oder -Buchse versehen und die Schalter angeschlossen sein.

Legen Sie Ihrer Kassettenplatine 10 DM (Schein oder Scheck) für Arbeits- und Versandkosten bei. Die Platine wird dann beim FSS richtig eingestellt und zurückgeschickt.

Da die Grundprogrammversion 4.3 gegenüber den ersten Versionen etliche Erweiterungen aufweist (Floppyroutinen, Blockbefehle im Editor, Textdruck, ...), können Sie Ihre alten EPROMS zum Umtausch einsenden. Bitte legen Sie Ihren EPROMS 35 DM für Handbuch und Kopierkosten bei.

Beim FSS werden Sie auch über Neuerungen und Weiterentwicklungen zum NDR-Klein-Computer informiert. **Außerdem gibt es bei uns noch etliche andere Programme für die CP/M-Version des Computers.**

Alles Wissenswerte erfahren Sie aus dem vierteljährlich erscheinenden, kostenlosen Katalog des FSS.

Die Lieferung der bestellten Ware erfolgt per Nachnahme (3.20 DM Nachnahmegebühr + 4 DM Porto und Verpackung). Wenn Sie per Verrechnungsscheck voraus-bezahlen (Zahlung gilt als Bestellung) sparen Sie Nachnahme und Porto.

Franzis-Verlag GmbH, Software-Service, Postfach 37 01 20, 8000 München 37

Franzis'

Rolf-Dieter Klein

Architektur und Sprache

Mehr Bits für die CPU

Schneller, höher, weiter, das ist das Gesetz des Fortschrittes. Mikroprozessoren, die pro Zeiteinheit mehr oder mächtigere Operationen durchführen können als Konkurrenz-Modelle, das sind die Entwurfsziele der Ingenieure in den Herstellerfirmen. Es ist einfach Tatsache, daß jemand, der ein Mikrocomputer-Professional sein will, deshalb unbedingt auch einen der modernen 16-Bit-Prozessoren kennenlernen muß.

Der NDR-Computer arbeitete bisher mit dem Mikroprozessor Z80. Dieser Mikroprozessor ist in der Industrie weit verbreitet, er ist eines der Arbeitspferde in der Mikrowelt. Der Z80 besitzt leider einen recht komplizierten Aufbau seiner Befehle. Es existieren sehr viele Ausnahmen und Sonderfälle, beispielsweise in der Behandlung der Registerpaare. Modernere Prozessoren besitzen demgegenüber durchsichtigere Befehle, die man durch Kombination mit Adressierarten aller möglichen Techniken zu einer großen Vielfalt ergänzen kann, ohne daß viele Ausnahmen und Regeln beachtet werden müssen. Ein Beispiel ist der Prozessor 68008, der nicht nur übersichtlichere, sondern auch mächtigere Befehle, wie eine eingebaute Multiplikation und Division, besitzt. Man benötigt bei ihm nur wenige Befehle, um komplizierte Vorgänge zu beschreiben. Das Schema, die Prinzipien, nach welchen ein solcher Baustein (oder auch ein ganzer Rechner) konstruiert ist, bezeichnet man als seine Architektur. Eine Computerarchitektur kann sowohl den elektrischen Aufbau einer Datenverarbeitungsanlage, als auch die Befehlsstruktur beschreiben. Der Begriff kann wie gesagt auf einen vollständigen Computer oder für Teile davon angewandt werden.
Bei den Mikrochips gehen geometrische und logische Architektur ineinander über (*Bild 1*).

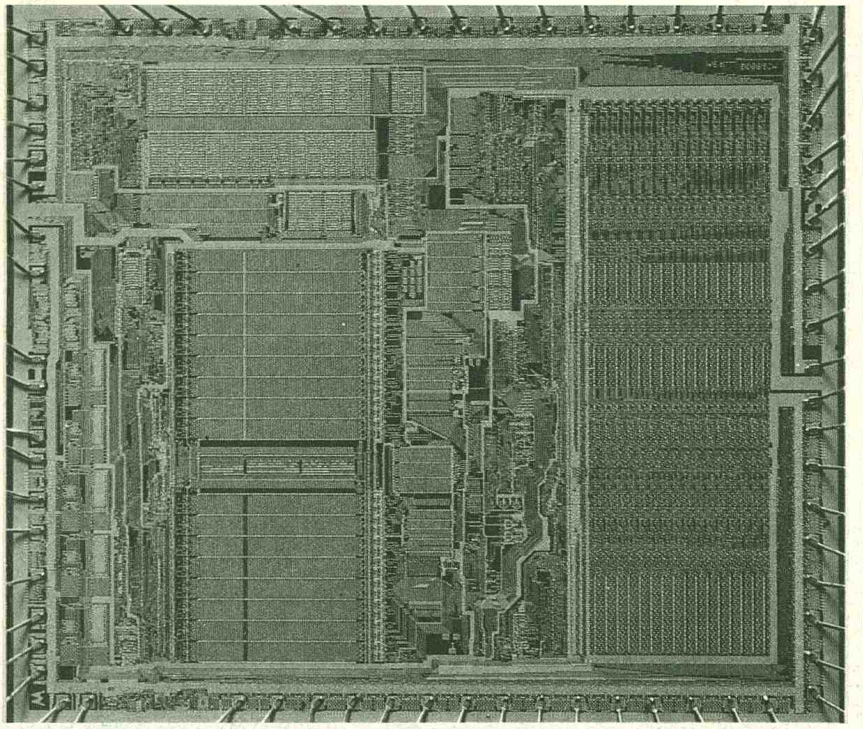


Bild 1. Das Layout des 68000 zeigt unglaublich viel Transistoren, nämlich 68000!

Der 68008

Wenn man zum Beispiel die Mikroprozessoren Z80 und 68008 miteinander vergleicht, so erkennt man, daß der 68008 eine feinere Struktur besitzt, denn auf dem Chip sind viel mehr Bauelemente (68000!) untergebracht als auf dem Z80. Man sagt, daß der 68008 eine höhere Integrationsdichte besitzt als der Z80.
Damit kann der Prozessor mehr leisten als der Z80. Zum Beispiel gibt es beim 68008 20 Adreßleitungen (*Bild 2*). Damit kann der 68008 bis zu 1 MByte Speicher ansprechen. Das sind 1 048 576 Speicherzellen mit je 8 Bit. Darin lassen sich sehr umfangreiche Programme und Datenmengen ablegen.
Der Prozessor 68008 besitzt noch einen großen Bruder, den Prozessor 68000. *Bild 3* zeigt die beiden Prozessoren im Gehäuse. Beide Prozessoren stimmen in der Programmierlogik überein. Der 68000 besitzt allerdings 23 Adreßleitungen und 16 Datenleitungen. Er ist ein 16-Bit-Rechner, der bis zu 16 MByte Speicher adressieren kann. Er benötigt wegen der Datenbusbreite von 16 Bit genau

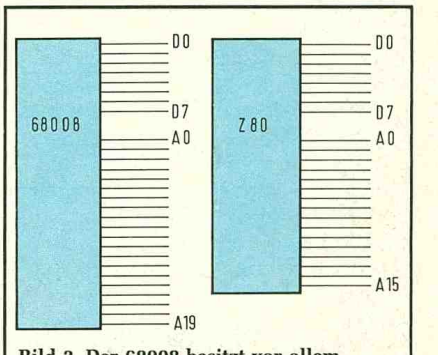


Bild 2. Der 68008 besitzt vor allem wesentlich mehr Adreßleitungen als der gute, alte Z80

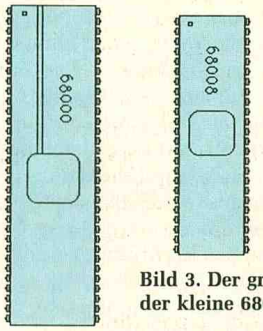


Bild 3. Der große und der kleine 68000

passenden Speicher, also Speicher, der ebenfalls 16 Bit parallel liefern kann. Da der Prozessor dann mit einem Atemzug gleich zwei Bytes vom Speicher holen kann, ist er noch ein Stück schneller als der 68008. Da aber beide Prozessoren intern gleichartig aufgebaut sind und der 68008 nur zweimal auf den Speicher zugreift, wo der 68000 mit einem Zugriff auskommt, der 68008 aber mit wesentlich preiswerterem und im NDR-Klein-System schon vorhandenen Speicher auskommt, wird er hier benutzt. Wer echte 16 Bit benutzen will, muß leider tiefer in die Tasche greifen.

Der Aufbau des 68008

Wenn man den Z80 mit dem 68008 vergleicht, so stellt man zunächst sehr viele Unterschiede fest. Die Signalleitungen des 68008 tragen andere Bezeichnungen, und die Befehle, wie auch die Register, sind Einsteigern zunächst sehr fremd. Allen Mikroprozessoren gemeinsam ist aber immer eine Gruppe von Signalleitungen, die zur Bedienung des Speichers und der Peripherie dienen. Dabei können jedoch die Bezeichnungen der einzelnen Leitungen stark voneinander abweichen.
Bild 4 zeigt die beiden ICs und deren Anschlußleitungen mit den Bezeichnungen. Man sieht, daß es bei beiden ICs ähnliche Gruppen gibt. Zum Beispiel gibt es bei beiden die Bus-Steuersignale oder die DMA-Leitung oder die Interrupt-Leitung sowie einen Daten- und Adreßbus. Man kann viele dieser Signale durch Gatter wechselseitig ineinander überführen. Manche sind auch schon von der Bedeutung her identisch. *Bild 5* zeigt ein Beispiel für den zeitlichen Ablauf verschiedener Signale. Die Bus-Steuersignale AS-Quer, DS-Quer und R/W-Quer des 68008 sind den Signalen MREQ-Quer, RD-Quer und WR-Quer des Z80 gegenübergestellt.

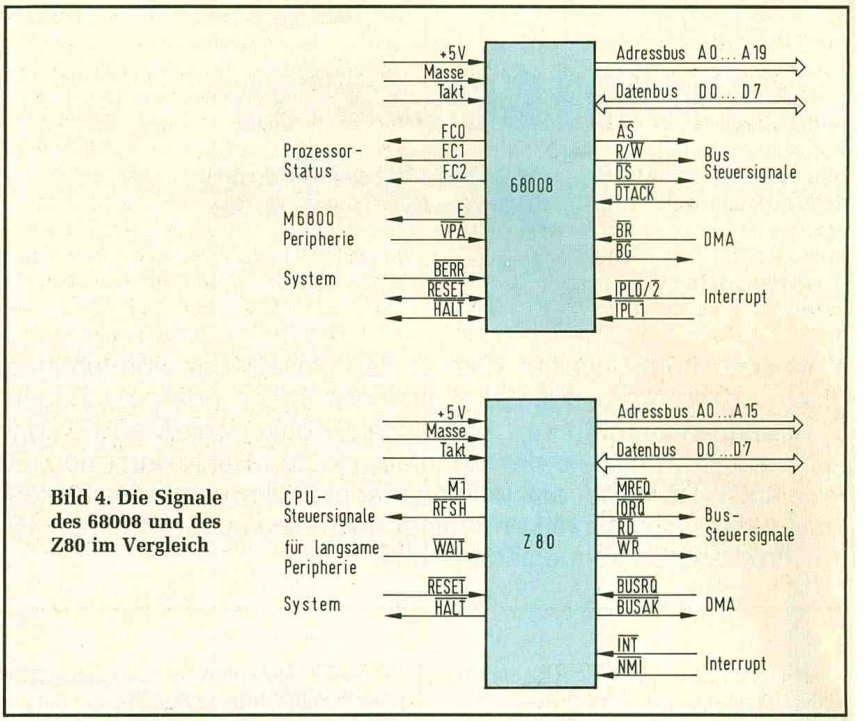


Bild 4. Die Signale des 68008 und des Z80 im Vergleich

Das Signal $\overline{AS}$ (Adreßstrobe) gibt an, daß eine Adresse auf dem Adreßbus gültig ist. Mit $\overline{DS}$ (Datastrobe) wird der Datenbus gesteuert. Beim Lesen legt der Prozessor 68000 das Signal R/$\overline{W}$ auf 1, beim Schreiben auf 0. Entsprechend heißt es Read/$\overline{Write}$-Signal.
Man kann aus den Signalen des 68008 durch logische Verknüpfung über TTL-Gatter die Signale des Z80 erzeugen. Zum Beispiel ergibt sich das $\overline{WR}$-Signal des Z80 aus der ODER-Verknüpfung des $\overline{DS}$-Signals mit dem R/$\overline{W}$-Signal. Denn ist eines der beiden Signale auf 1, so ist auch das $\overline{WR}$-Signal auf 1, oder anders ausgedrückt, nur wenn beide Signale (R/$\overline{W}$ und $\overline{DS}$) auf 0 sind, so ist auch $\overline{WR}$ auf 0.
Die aus den Signalen des 68008 abgeleiteten „Z80-Steuerimpulse" weisen noch kleine zeitliche Unterschiede gegenüber dem Original auf, doch dies ist für die logische Funktion, die für das Arbeiten mit dem Speicher allein maßgeblich ist, nicht von Bedeutung. Ein wichtiger Unterschied zwischen den beiden Prozessoren besteht darin, daß der 68008 kein besonderes Signal für die Adressierung der Peripherie besitzt. Beim Z80 leistet dies das $\overline{IORQ}$-Signal. Beim 68008 gibt es so ein Signal nicht. Bei ihm muß man einen Adreßbereich decodieren und mit dem Signal $\overline{AS}$ verknüpfen, wenn man Peripherie ansprechen will. Im Bild 5 erkennt man, daß das $\overline{MREQ}$-Signal direkt dem $\overline{AS}$-Signal folgt. Wenn man nun das Signal $\overline{AS}$ zusätzlich mit Adressen verknüpft, so daß das daraus abgeleitete $\overline{MREQ}$ nur dann erscheint, wenn der Adreßbereich für den Speicher angewählt ist und in einer weiteren Verknüpfung einen anderen Adreßbereich mit $\overline{AS}$ so verknüpft, so kann man das zweite Signal als $\overline{IORQ}$ verwenden.
Man sieht also, daß auch von der Architektur her völlig verschiedene Mikroprozessoren, doch im wesentlichen einen gemeinsamen Kern haben. Sie müssen mit einem Speicher arbeiten, aus dem sie Befehle und Daten erhalten, und sie müssen mit Peripherie arbeiten. Und dazu benötigen sie einmal Daten und Adressen, sowie Steuersignale, die man alle auf die Operationen „Auswahl" und „Lesen" oder „Schreiben" zurückführen kann. Daher gelingt es im Prinzip immer, jeden Mikroprozessor an die diskutierten Signale anzugleichen.

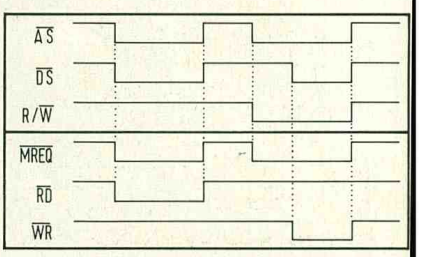


Bild 5. Das Timing muß kompatibel gemacht werden

So arbeitet die neue CPU

Der Takt wird wieder mit dem IC 74LS04 erzeugt (IC5, Pin 1...6 im Schaltplan, *Bild 6*). Die Quarzfrequenz beträgt jetzt 8 MHz. Man kann den Takt bei Bedarf halbieren, dazu ist in der Schaltung noch ein IC 74LS74 vorgesehen, doch normalerweise wird es nicht verwendet. Es ist eigentlich dafür gedacht, mit einem 16-MHz-Quarz einen symmetrischen 8-MHz-Takt zu erzeugen.
Die Reset- oder Startlogik wird durch das IC 555 (IC1) gebildet. Dieses IC ist als Monoflop geschaltet. Man hätte auch ein 74121 verwenden können, jedoch ist das IC 555 in seinen Abmessungen kleiner und liefert sehr präzise Zeitsignale, auch bei großer Länge.
Der HALT- und RESET-Eingang der CPU erhalten beide das RESET-Signal nach dem Spannungseinschalten oder wenn man die Reset-Taste drückt. Da RESET und HALT auch Ausgänge sein können, werden sie über Gatter mit offenen Kollektorausgängen (7405, IC2) angesteuert. Der RESET-Ein/Ausgang der CPU ist zudem so verschaltet, daß er auch einen Reset am Bus auslösen kann. Dazu gibt es beim 68008 einen eigenen Befehl.
Die beiden Ein-Aus-Vier-Decoder im 74LS139 (IC10) haben die Aufgabe, aus den 68008-Steuersignalen die Z80-ähnlichen Signale für den Bus zu erzeugen. *Bild 7* zeigt die Wahrheitstabelle für das IC. Das IORQ-Signal wird immer dann erzeugt, wenn auf den Adreßbereich Fxxxx zugegriffen wird. Beim Zugriff auf den Adreßbereich 0xxxx bis Exxxx wird das MREQ-Signal erzeugt. Damit stehen theoretisch 65536 Adressen für die Peripherie zur Verfügung. Auf den NDR-Klein-Standard-Baugruppen werden jedoch nur die Adreßleitungen A0 bis A7 zur Decodierung herangezogen. Man kann also real nur 256 verschiedene Adressen vergeben, wie es auch beim Z80 vorgesehen ist. An die Decoder 74LS139 ist zusätzlich das Signal VPA geführt. Es dient, wie hier verschaltet, der Interrupt-Steuerung beim 68008. Es wird der nicht vektorisierte Interrupt verwendet. Doch davon später mehr. Die Bausteine IC9, IC11, IC12 und IC13 dienen der Pufferung der Daten- und Adreßleitungen.

Wartezyklen: einstellbar

Eine Besonderheit stellt das IC 74LS164 (IC4) auf der CPU-Karte dar. Damit lassen sich sogenannte Warte-Zyklen einfügen. Wenn man eine Brücke bei JMP1 einsetzt, wird jeder Zugriff auf den Speicher oder die Peripherie verlängert. Der kürzeste Wartezyklus ist über die Brücke

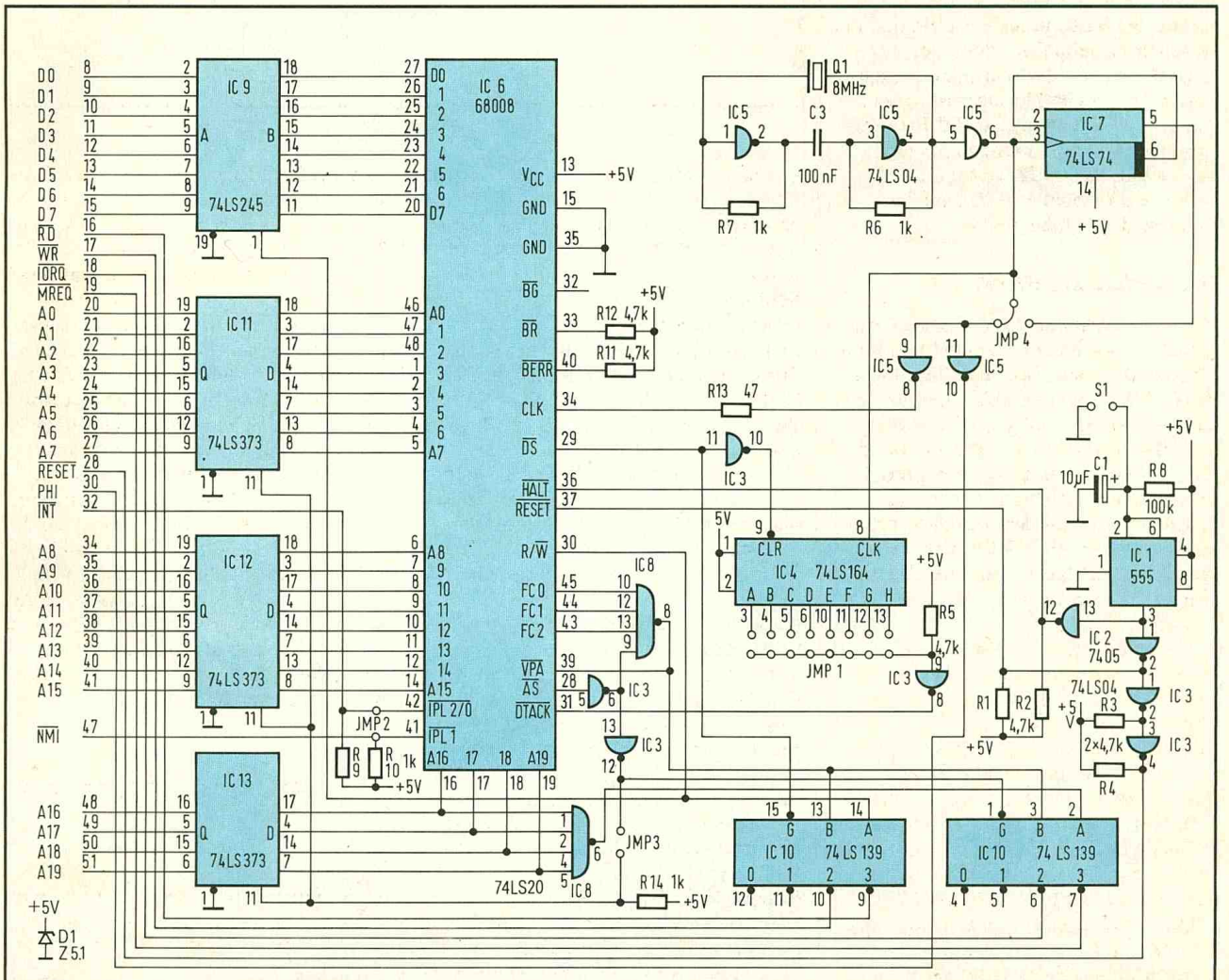


Bild 6. Das Schaltbild der Karte CPU68K

zum Eingang A, der längste über den Eingang zur Brücke H eingestellt. Ganz ohne Brücke erfolgen alle Zugriffe mit maximaler Geschwindigkeit.
Die CPU besitzt zur Steuerung einen Eingang mit der Bezeichnung $\overline{\text{DTACK}}$. Liegt er auf 0 V, was ohne Brücke über R5 und IC3 (Inverter) erzwungen wird, so wird jeder Zugriff mit maximaler Geschwindigkeit durchgeführt. Wenn man eine der genannten Brücken einsetzt, ist das IC 74164, ein Schieberegister, für die $\overline{\text{DTACK}}$-Erzeugung verantwortlich. Das Signal DS am CLR-Eingang des Schieberegisters sorgt dafür, daß alle Ausgänge des Schieberegisters auf 0 V liegen, wenn ein Zugriff erfolgen soll. Über die Brücke gelangt das Null-Signal an den Inverter und am Ausgang des Inverters liegt jetzt ein 1-Signal, das an den $\overline{\text{DTACK}}$-Eingang der CPU gelangt. Die CPU wartet nun so lange, bis das $\overline{\text{DTACK}}$-Signal am Eingang wieder auf 0 geht. Dies geschieht nun in Abhängigkeit von der Brückenstellung. Denn unmittelbar nachdem das $\overline{\text{DS}}$-Signal auf 0 geht, um einen Zugriff zu signalisieren, wird eine 1, die an den Pins 1 und 2 dauernd anliegt, durch den Schiebetakt an CLK der Reihe nach von Ausgang A nach Ausgang H durchgeschoben. Wenn die Brücke zum Beispiel an D angeschlossen wurde, kommt das 1-Signal erst nach dem vierten Schiebetakt am Ausgang D an. Also wird der $\overline{\text{DTACK}}$-Eingang der CPU nach dem vierten Takt auf 0 gehen und die CPU wird erst jetzt den Zugriff abschließen.
Wenn man $\overline{\text{DTACK}}$ auch außerhalb der CPU verwenden will, so muß man eine kleine Zusatzschaltung nach *Bild 8* verwenden. Diese Schaltung erzeugt aus einem $\overline{\text{WAIT}}$-Signal das $\overline{\text{DTACK}}$-Signal. Man kann dann durch andere Karten die CPU anhalten, wenn diese mit einem $\overline{\text{WAIT}}$-Signal anzeigen, daß sie noch Zeit benötigen. Die Schaltung verwendet zwei freie Gatter auf der CPU, sowie einen extra Widerstand. Die Schaltung wird in zukünftigen Versionen der CPU-Baugruppe schon enthalten sein. Sie wird zum Beispiel beim Betrieb mit dynamischen Speichern verwendet. Zum Test der CPU-Karte und für den normalen Betrieb muß man die Schaltung jedoch nicht aufbauen.
Bild 9 zeigt den Bestückungsplan der Leiterplatte und *Tabelle 1* die Stückliste. Auf der Leiterplatte gibt es vier Gruppen von Brücken (JMP). Mit JMP1 stellt man die Zahl der Wartezyklen ein. Diese Brücke solle man vor dem ersten Versuch auf Position 2 stellen. Die Brücke JMP2 wird verwendet, um die Interruptbetriebsart umzustellen. Sie bleibt offen.

$\overline{G}$	B	A	0	1	2	3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

Bild 7. Die Wahrheitstafel des Bausteines 74LS139

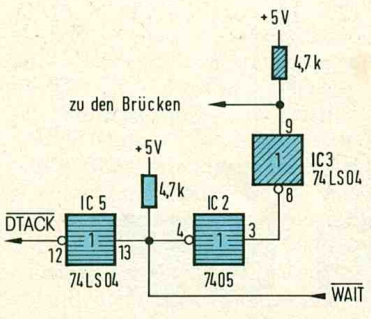


Bild 8. Das ist die Zusatzschaltung (die schon auf der Platine im Layout vorbereitet ist), mit der die CPU auf $\overline{\text{WAIT}}$ reagieren kann

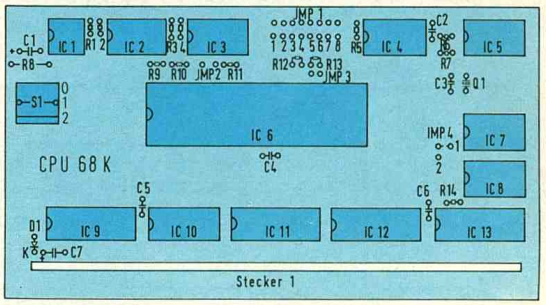


Bild 9. Der Bestückungsplan zur CPU-Karte mit 68008

Tabelle 1. Die Stückliste

Stück	Benennung	
1	IC1	NE 555
1	IC2	74 05
1	IC3	74 LS 04
1	IC4	74 LS 164
1	IC5	74 LS 04
1	IC6	CPU 68008
1	IC7	74 LS 74
1	IC8	74 LS 20
1	IC9	74 LS 245
1	IC10	74 LS 139
3	IC11,12,13	74 LS 373
1	SO 8	8polige IC-Fassung
6	SO 14	14polige IC-Fassung
4	SO 20	20polige IC-Fassung
2	SO 24	24polige IC-Fassung (statt der 48poligen)
7	R1,2,3,4,5,11,12	4,7 kΩ
5	R6,7,9,10,14	1 kΩ
1	R8	100 kΩ
1	R13	47 Ω
2	C1,7	10 µF
5	C2,3,4,5,6	100 nF
1	S1	Taster
1	Q1	Quarz 8 MHz
1	D1	Zenerdiode 5,1 V
1	ST 1	18- und 36polige Steckerleiste
1	JMP1	Stiftreihe 8×2
1		Shuntstecker
1	Leiterplatte CPU 68 K mit Lötstopplack	

Die Brücke wird erst für einen der späteren Versuche benötigt. Die Brücke JMP3 bleibt ebenfalls offen. Setzt man sie ein, so wird das Zeitverhalten der Adreßleitungen verändert. Normalerweise wird diese Option nicht benötigt. JMP4 ist durch eine Leiterbahn schon so eingestellt, daß das Teiler-IC zunächst übergangen wird und der Takt eins zu eins weitergeleitet wird.

Der Aufbau der Baugruppe

Folgende Schritte sollten Sie einhalten:

1. Einbau aller Sockel und passiven Bauteile.
2. Einsetzen aller ICs, bis auf die CPU.
3. Die Baugruppe CPU 68K mit der Versorgungsspannung verbinden.
4. Messen an Pin 34 der CPU-Fassung. Dort muß ein 8-MHz-Takt ankommen.
5. Messen an Pin 36 und an Pin 37 der CPU-Fassung. Wenn man den RESET-Taster drückt, so muß der Pegel von +5 V auf 0 V wechseln.
6. Einsetzen der CPU.

Den restlichen Test kann man erst durchführen, wenn man auch die ROA64-Baugruppe aufgebaut hat.

Die Speicherbaugruppe ROA64

Bild 10 zeigt die Schaltung. Auf der Baugruppe können bis zu 64 KByte Speicher untergebracht werden. Dabei ist es möglich, sowohl EPROMs als auch RAM-Bausteine zu mischen, denn diese sind heute anschlußkompatibel. Das IC 74LS245 (IC9) dient zur Trennung vom Bus und als Bustreiber oder -Puffer. Das IC10 ist der Decoder, der die einzelnen Bausteine auf der Baugruppe auswählt und der Baustein 74LS85 (IC12) dient der Adressierung der Karte. Beim 68008 kann man nämlich, wegen des 1 MByte Adreßbereiches, 16 solcher Karten verwenden. Die Adresse der Karte wird mit den Brücken bei JMP2 eingestellt. Dabei entspricht eine eingesetzte Brücke einer logischen 0 auf der im Schaltbild gegenüberliegenden Adreßleitung. Mit der Brücke JMP1 kann man noch einstellen, ob man den 6116-Baustein verwenden will. Dieser RAM-Baustein besitzt nur 24 Pins und an Pin 23 (zählweise als ob 28 Pins) benötigt er anstelle der Adresse A11 das Signal $\overline{WR}$. *Bild 11* zeigt den

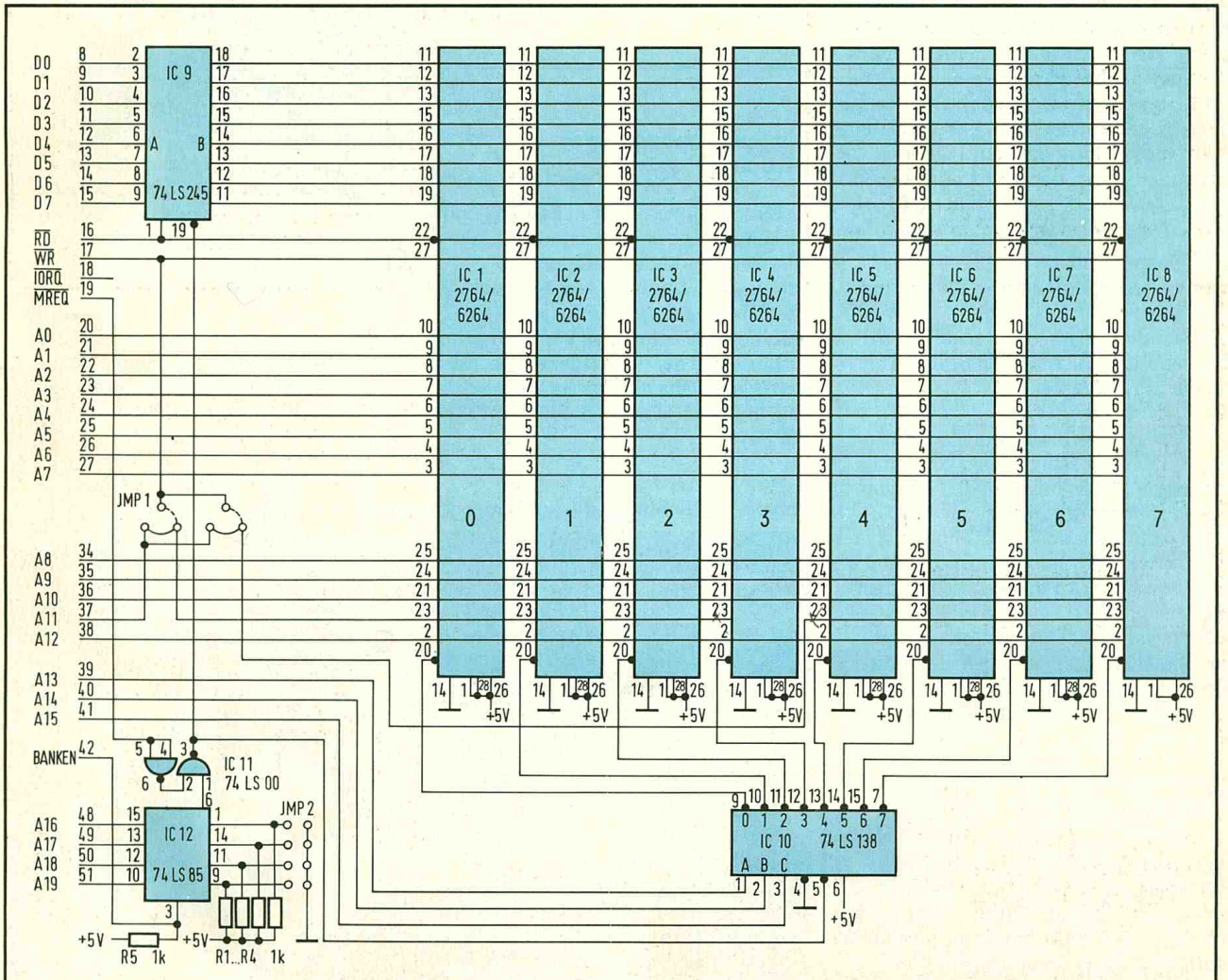


Bild 10. Das Schaltbild der Speicherkarte mit statischen RAM-Bausteinen. Es können in jeden Sockel auch EPROMs eingesteckt werden

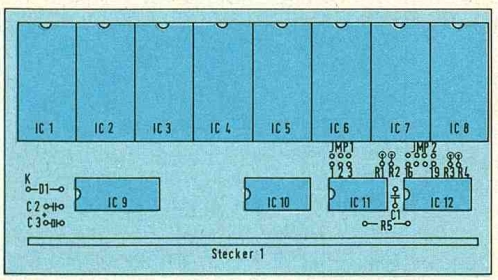


Bild 11. Der Bestückungsplan zur Speicherbaugruppe

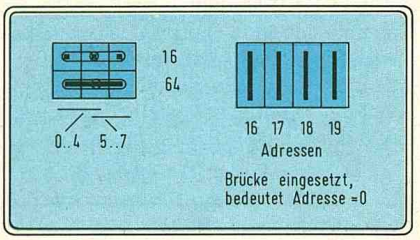


Bild 12. So werden die Brücken auf der Speicherbaugruppe eingestellt

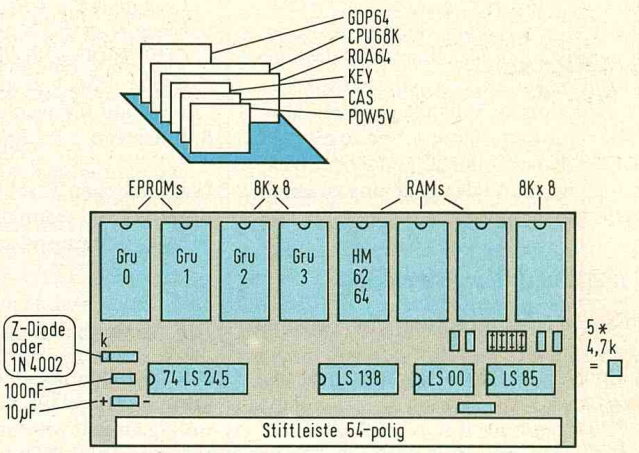


Bild 13. Die Einordnung der Karten für ein vollständiges 68008-System. Für den Betrieb müssen die vier EPROMs mit dem Grundprogramm und mindestens ein RAM-Baustein auf der ROA64 eingesteckt sein

Bestückungsplan und *Tabelle 2* die Stückliste. *Bild 12* zeigt die Einstellung der Brücken, wie sie für das Grundprogramm im Betrieb mit dem 68008 benötigt werden. Die Karte wird auf den Adreßbereich 00000 bis 0FFFF gelegt, daher werden alle Brücken bei A16 bis A19 eingesteckt. Die Brücke JMP1 wird unverändert gelassen, denn auf der Leiterplatte sind die richtigen Brücken schon vorverdrahtet.

Bild 13 zeigt die Einordnung der Karte ins System. Die Baugruppen können in beliebiger Reihenfolge auf den Bus gesteckt werden, das Bild ist nur ein Beispiel. Man kann noch die kleine Busplatine verwenden, jedoch müssen jetzt die Buchsenreihen für die längeren System-Karten eingelötet werden. Wer Größeres vorhat, sollte sich gleich einen Bus mit mehr Steckplätzen besorgen, damit für Erweiterungen Platz ist (man kann auch den kleinen Bus einfach verlängern, indem man weitere kleine Busse anlötet).

Die EPROMs des Grundprogramms werden so eingesetzt, wie es Bild 13 zeigt. Man achte auf die richtige Lage der Nase bei dem Einbau der ICs. Der Baustein HM-6264, der übrigens auch manchmal mit anderer Bezeichnung geliefert wird, ist ein 8-K×8-RAM-Baustein. Man kann dort 8192 Bytes unterbringen. Wer größere Programme eingeben will, sollte sich sofort einen weiteren Baustein anschaffen und an der nächsten Stelle (IC6) einbauen.

Die Speicherbaugruppe wird aufgebaut

1. Einlöten aller passiven Bauteile und Fassungen. Die eingezeichnete DIODE oder Zehner-Diode dient nur zum Schutz des Computers gegen Verpolung oder Überspannung, sie hat keine direkte schaltungstechnische Bedeutung.
2. Einsetzen der ICs. Widerstände, die sich in der Nähe der Brücke JMP2 befinden, muß man stehend einlöten (R1, R2, R3, R4).
3. Einsetzen der EPROMs. Bitte Beschriftung und Polung beachten.
4. Einsetzen des RAM-Bausteins. Achtung, es handelt sich um einen CMOS-Baustein, der empfindlich auf elektrostatische Entladungen reagiert. Daher das Einsetzen möglichst auf dem schwarzen MOS-Schaumstoff durchführen: Platine auf den Schaumstoff legen, Speicherbaustein mit seinem Schutzschaumstoff ebenfalls darauflegen, dann mit einer Hand Platine festhalten, mit der anderen den Baustein einsetzen.

Tabelle 2. Die Stückliste zur Speicherbaugruppe

Stück	Benennung	
1	IC1	EPROM 2764 Assembler 0 oder Pascal 0
1	IC2	EPROM 2764 Assembler 1 oder Pascal 1
1	IC3	EPROM 2764 Assembler 2 oder Pascal 2
1	IC4	EPROM 2764 Assembler 3 oder Pascal 3
1	IC5	RAM 6264
1	IC6	RAM 6264 oder EPROM 2764
1	IC7	RAM 6264 oder EPROM 2764
1	IC8	RAM 6264 oder EPROM 2764
1	SO 20	20polige IC-Fassung
1	SO 14	14polige IC-Fassung
2	SO 16	16polige IC-Fassung
8	SO 28	28polige IC-Fassung
5	R1...R5	1 kΩ (oder 4,7 kΩ)
2	C1, C2	100 nF
1	C3	10 µF
1	D1	Zenerdiode 5,1 V
1	St 1	36- und 18polige Steckerleiste
1	JMP1	Stiftreihe 2 × 4
4		Shuntstecker
1	Leiterplatte ROA64 mit Lötstopplack	

5. Einsetzen der vier Brücken bei 16 bis 19 auf die Stiftleisten.
6. Aufbau der Gesamtanordnung nach Bild 15 überprüfen.
7. Karte auf den Bus stecken.
8. Einschalten der Versorgungsspannung. Auf dem Bildschirm muß sich das Grundprogramm melden.
9. Meldet sich das Grundprogramm nicht, so entfernt man die Baugruppen GDP64, KEY und CAS. Mit dem Prüfstift oder einem Oszilloskop wird die Leitung $\overline{\text{IORQ}}$ beobachtet. Dort müssen Pulse erscheinen. Wenn nicht, so liegt der Fehler wahrscheinlich auf einer der Baugruppen CPU68K oder ROA64. Sind die Pulse da, so liegt der Fehler bei der GDP64-Baugruppe (KEY und CAS nur dann, wenn Kurzschlüsse vorliegen). Wenn die Schaltung nicht funktioniert, sollte man zuerst alle Lötstellen prüfen, denn dies ist immer die Fehlerquelle Nummer eins.

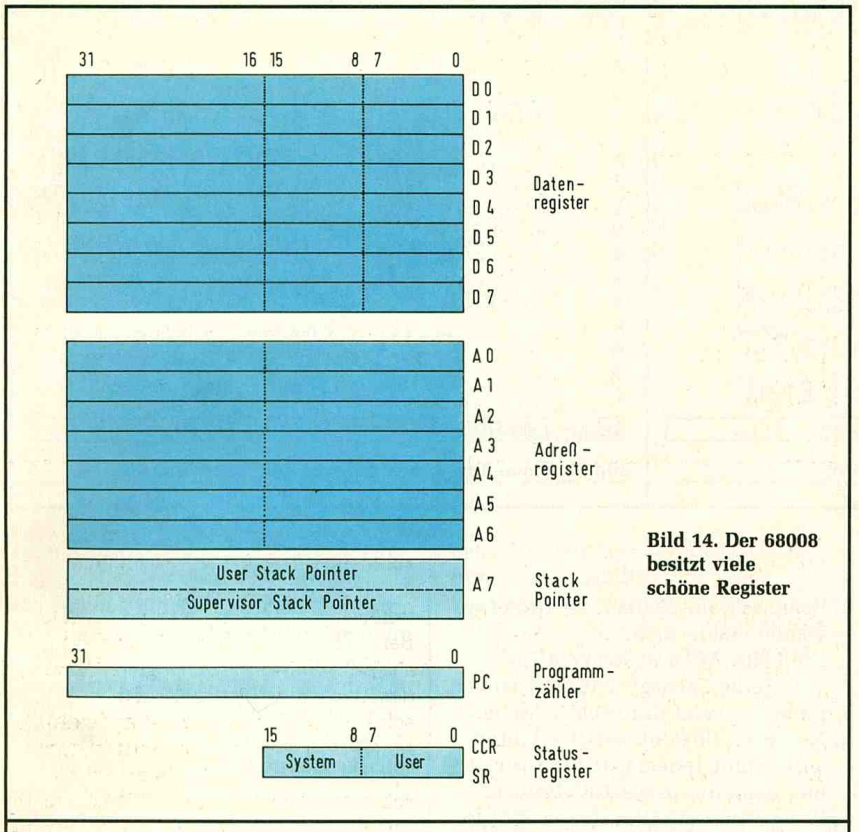


Bild 14. Der 68008 besitzt viele schöne Register

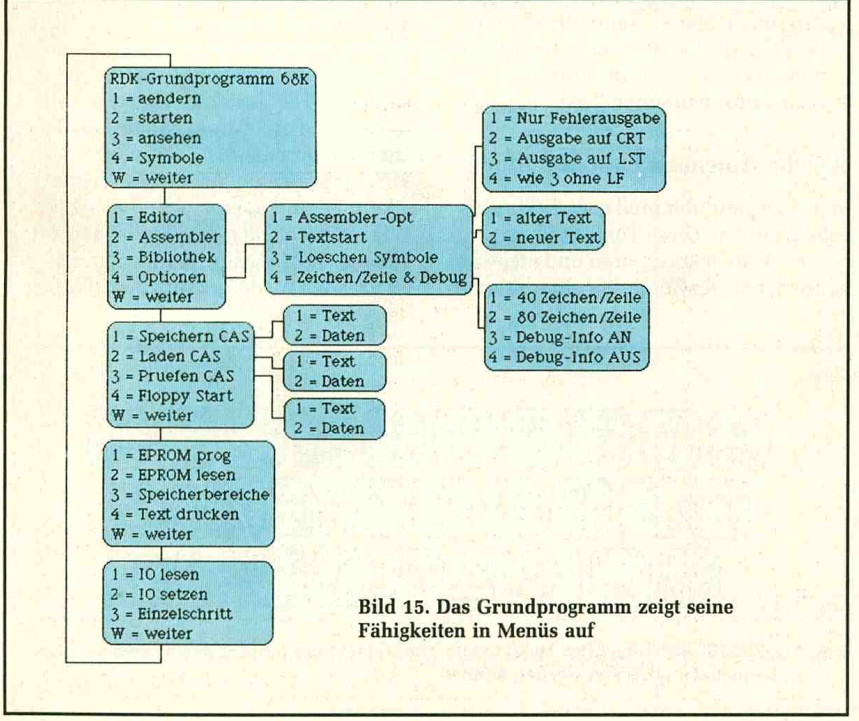


Bild 15. Das Grundprogramm zeigt seine Fähigkeiten in Menüs auf

Die Registerstruktur des 68008

Der Mikroprozessor 68008 besitzt eine Vielzahl von Registern. *Bild 14* zeigt eine Übersicht. Der 68008 besitzt eine Reihe von Datenregistern, in denen vorzugsweise Rechenoperationen durchgeführt werden, sowie eine Anzahl von Adreßregistern, die normalerweise für die Adressierung von Speicherzellen verwendet werden. Alle diese Register sind 32 Bit breit, das heißt, sie können 32 Bit breite Dualzahlen aufnehmen. Die Datenregister kann man zusätzlich in 8 oder 16 Bit breite Einheiten unterteilen, wenn man nur mit Bytes oder mit 16-Bit-Worten arbeiten will. Die Adreßregister können ebenfalls manchmal als 16-Bit-Größen verwendet werden, wenn man nur bestimmte, eingeschränkte Speicherbereiche ansprechen will. Eine besondere Rolle spielt das Register A7, das in zwei Hälften geteilt ist. In Wirklichkeit besteht es aus zwei Registern, von welchen aber immer nur eines zugänglich ist. Der 68008 hat nämlich zwei Betriebsarten. Bei der sogenannten Benutzerbetriebsart (User-Mode) sind nicht alle Befehle verwendbar. Das ist so, weil dann in einem Betriebssystem bestimmte Operationen für den Anwender gesperrt werden können. In der Betriebsart „System" (System-Mode) sind dem Programmierer alle Befehle zugänglich. Diesen Modus verwenden wir im Grundprogramm.
Das Register A7 ist der Stackpointer. Es wird denen, die schon einmal mit dem Z80 gearbeitet haben, als Register SP bekannt sein. Das Register PC ist der Programmzähler. Er ist ebenfalls 32 Bit breit, obwohl nur 20 Adreßleitungen beim 68008 herausgeführt sind.
Dann gibt es noch ein Statusregister, das in zwei Hälften geteilt ist. In den Bits 0 bis 7 sind die Bedingungsflags untergebracht und in den Bits 8 bis 15 wichtige Systemeinstellungen, wie zum Beispiel die Betriebsart System oder User. Wie man mit den Registern arbeitet, das zeigt

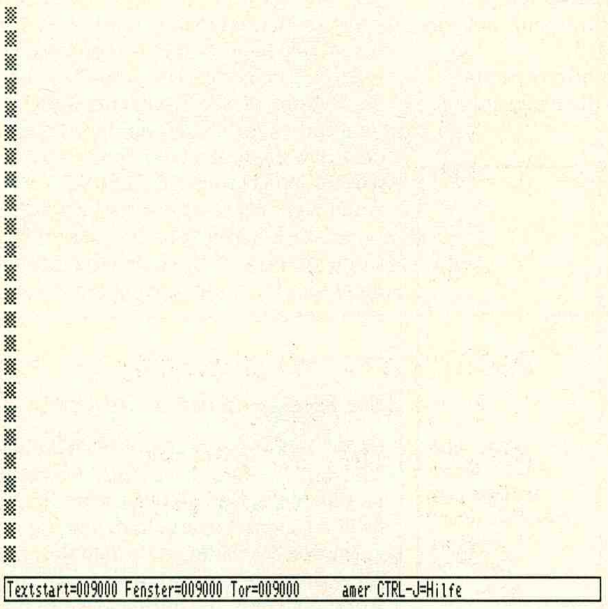


Bild 16. Das zeigt sich nach Aufruf des Editors

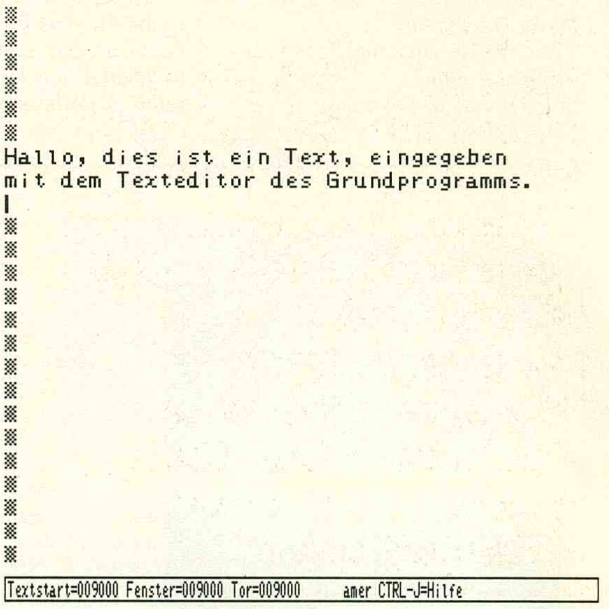


Bild 17. Dieser Text wurde mit dem Editor geschrieben

sich erst im Laufe der Folgen, dieser kurze Überblick soll fürs erste einmal genügen.

Das Grundprogramm für den 68008

Der Cursor blinkt sicher schon ungeduldig auf Ihrem Bildschirm, wenn Ihr Gerät nach dem Einschalten nicht gleich den Elektroniktod gestorben ist. Das Grundprogramm für den 68008, das jetzt Ihre Eingaben erwartet, ist ähnlich aufgebaut wie das für den Z80, nur daß es hier viel mehr Menüpunkte gibt. *Bild 15* zeigt eine Übersicht. Durch die Tasten W und CR kann man die Bildschirme, die untereinander liegen, aufrufen. Die Untermenüs, von welchen nur die Optionen aus dem zweiten Bildschirm etwas genauer gezeigt sind, wählt man durch Tippen der zugeordneten Ziffer an. Wir beschäftigen uns hier zunächst nur mit dem Editor, der im zweiten Menü angewählt werden kann. Dazu geht man wie folgt vor:

1. Spannung einschalten, das Grundprogramm meldet sich.
2. Die Taste „W" und dann die Taste „CR" (oder Carriage Return) betätigen.
3. Dann erscheint das nächste Menü.
4. Nun wird die Funktion „1 = Editor" ausgewählt, indem man die Taste „1" und dann die Taste „CR" drückt.
5. Es erscheint ein Bild wie in *Bild 16*. Am linken oberen Rand blinkt ein helles Feld, der Cursor, der ja auch schon vom „Weiter"-Feld bekannt ist.
6. Nun kann man einen Text eingeben.

Welche Aufgaben hat ein Editor?

In jeden Computer muß man Texte eingeben können. Diese Texte muß man natürlich auch korrigieren und ergänzen können, ehe der Computer sie weiterverarbeitet. Bei unserem Editor kann man mit den Buchstabentasten Texte eingeben, mit der Taste CR kann man in die nächste Zeile gehen, mit der Leertaste kann man Leerzeichen in den Text eingeben, wie *Bild 16* zeigt. Um Korrekturen durchführen zu können, benötigt man zusätzliche Zeichen auf der Tastatur. Die beiden wichtigsten Tasten dazu zeigt *Bild 18*. Durch Drücken der Taste CTRL zusammen mit einer normalen Taste kann man sogenannte Steuercodes erzeugen. Die Tastatur erzeugt dabei Bitmuster, die nichtdruckenden ASCII-Zeichen entsprechen.
Ein solches Zeichen kann der Rechner erkennen und entsprechend reagieren. Wie bei der SHIFT-Taste muß man zuerst die CTRL-Taste, dann, während CTRL noch niedergedrückt ist, die Zei-

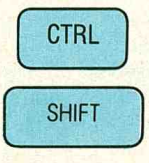


Bild 18. Zwei Beispiele für Steuertasten

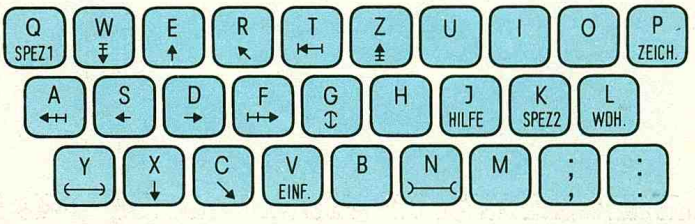


Bild 19. Der Texteditor besitzt viele Funktionen, die mit den Steuertastensequenzen ausgelöst werden können

```
REV 4.2 -1-
↑E=Zeile hoch      ↑X=Zeile runter
↑S=Zeichen links   ↑D=Zeichen rechts
↑A=Wort links      ↑F=Wort rechts
↑Z=Zeile auf       ↑W=Zeile ab
↑Y=Zeile loeschen  ↑N=Zeile einfuegen
↑T=Loeschen rechts
↑R=Seite zurueck   ↑C=Seite vor
↑V=Einfuegemode    ↑P=Zeichensatz
↑QC=an Textende    ↑QR=an Textanfang
↑QF=suchen         ↑QA=ersetzen
↑G,DEL=Zeichen loeschen
```

Bild 20. Auf Hilferufe erscheint dieses Bild auf dem Bildschirm...

```
REV 4.2 -2-
↑L=wiederhole ↑QF,↑QA

↑KX=ENDE

↑KB=Block Anf.  ↑KK=Block Ende
↑KY=Block loeschen
↑KV=verschieben ↑KC=kopieren
↑KR=lesen       ↑KW=speichern
↑KH=Marken loeschen
```

Bild 21. ...und dieses Bild schließt sich an

chen-Taste betätigen. Die Taste CR zum Beispiel liefert so einen Steuercode. Man kann anstelle von CR auch die Tasten CTRL und M drücken.
Bild 19 gibt eine erste Übersicht über die Steuerfunktionen. Wenn man die Taste CTRL zusammen mit einem Buchstaben drückt, unter dem ein Zeichen geschrieben ist, so wird die zugehörige Editorfunktion ausgeführt.
Zum Beispiel kann man den Cursor mit den Tasten S, D, E und X positionieren. Im Bild sind unter den Tasten Pfeile gezeichnet, die die Richtung angeben, in der sich der Cursor bewegt, wenn man die Tasten zusammen mit der Taste CTRL drückt. Sie können also den Cursor nach links, rechts, oben und unten bewegen.
Die vier Positionierfunktionen arbeiten aber immer nur auf gültigem Text. Am Textende oder überall dort, wo kein Text mit einer gewünschten Bewegung mehr überstrichen werden kann, bleibt der

```
Hallo dies ist ein Testtext.
Block verschieben ist bei REV 4.2 möglich.
▶Block der bearbeitet werden soll◀

Textstart=009000 Fenster=009000 Tor=009000 einf deut CTRL-J=Hilfe
```

Bild 22. Mit den Markierungen können Blöcke definiert werden, die im Ganzen behandelt werden können

```
Block der bearbeitet werden soll
▶Block der bearbeitet werden soll◀

Textstart=009000 Fenster=00904A Tor=00904A einf deut CTRL-J=Hilfe
```

Bild 23. Ein Block, der mit CTRL-KC behandelt wurde

Cursor einfach stehen. Hier eine wichtige Besonderheit. Ganz oben im Bild 17 und auch unten befinden sich links schraffierte Felder von der Größe eines Buchstabens. Sie dienen der Kennzeichnung, daß in diesen Zeilen kein Text steht. Wenn man einen Text in die Zeilen schreiben will, so muß man das schraffierte Feld einfach überschreiben und dadurch löschen. Will man eine leere Zeile einfügen, so muß man das schraffierte Feld durch das Leerzeichen (Leertaste drücken) überschreiben. Wenn man versehentlich, z. B. durch Positionierung des Cursors, rechts neben einem solchen Feld schreibt, so wird der Text später wieder gelöscht.

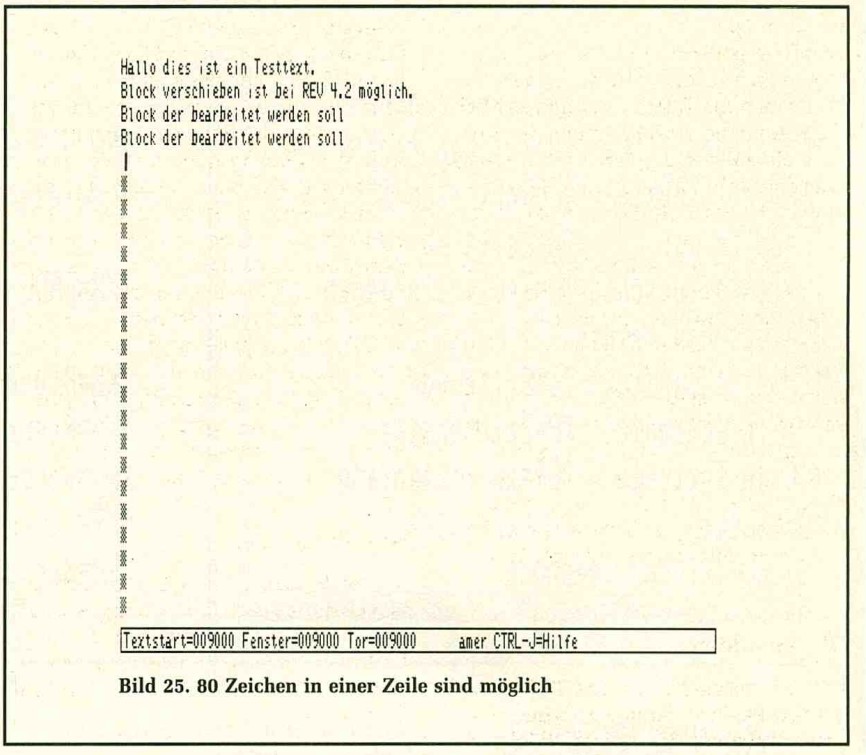


Bild 25. 80 Zeichen in einer Zeile sind möglich

Viele Funktionen

Es gibt viele Steuerzeichen, die komplizierte Funktionen auslösen. CTRL-A setzte den Cursor ein Wort nach links und CTRL-F ein Wort nach rechts. Mit CTRL-W und CTRL-Z kann man das Bildfenster verschieben, mit CTRL-R und CTRL-C kann man Seiten umblättern. Mit CTRL-G löscht man ein Zeichen rechts vom Cursor, mit CTRL-Y löscht man eine ganze Zeile. Mit CTRL-N fügt man eine Zeile neu ein. Auf der Tastatur gibt es eine Sondertaste mit der Beschriftung DEL. Damit kann man ein Zeichen links vom Cursor löschen, der Cursor rückt anschließend nach links.
Wenn man CTRL-V drückt, so erscheint im unteren Bildfenster der Text EINF. Wenn man jetzt Texte eingibt, so werden sie an der aktuellen Cursorstelle in den alten Text eingefügt. Wenn man erneut CTRL-V drückt, verschwindet der Schriftzug EINF wieder aus dem Bildfenster. Wenn man jetzt Zeichen eingibt, so wird ein evtl. vorhandener Text überschrieben.
CTRL-P besitzt ebenfalls eine Umschaltfunktion. Man kann damit den deutschen Zeichensatz einschalten, um die Buchstaben „Ä", „Ö", „Ü" und „β" zu erhalten. Bei der neuen Version des Grundprogramms (4.2) kann man sogar gemischte Eingaben durchführen, mit deutschen und amerikanischen Zeichen zugleich. Übrigens kann man durch die Tasten CTRL-J ein Hilfemenü auf dem Bildschirm bekommen (*Bild 22* und *23*), mit dem man sich über alle Befehle informieren kann.
Die geschilderten Kommandos reichen noch nicht aus. Es gibt CTRL-Tastenfolgen, die durch Kombination mit weiteren Buchstaben eine Bedeutung erhalten. Wenn man zum Beispiel die Tasten CTRL-Q drückt, so erwartet das Grundprogramm eine weitere Taste. Dies muß ein Großbuchstabe sein (bei der neuen Grundprogramm-Revision 4.2 kann man auch Kleinbuchstaben oder CTRL mit dem Buchstaben verwenden).

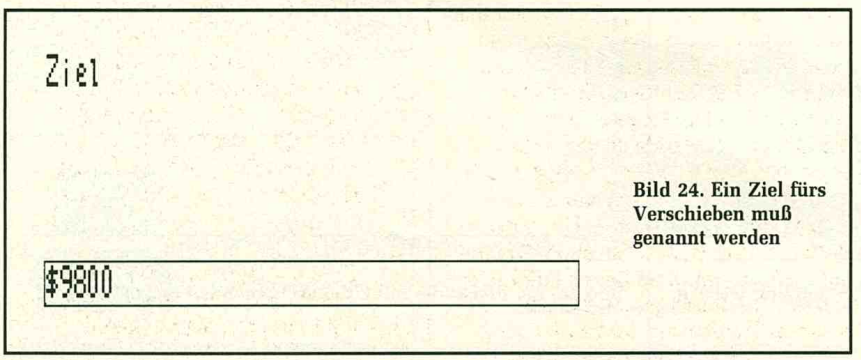


Bild 24. Ein Ziel fürs Verschieben muß genannt werden

Wenn man nach CTRL-Q die Tasten SHIFT-C drückt, kommt man an das Ende des Textes in Arbeit. Mit den Tasten SHIFT-R kommt man an den Anfang des Textes. Mit SHIFT-F kann man nach Wörtern suchen. Dazu erscheint ein Menü, das ein Eingabefeld zeigt. Dort muß man das Suchwort oder den interessanten Teil eines gesuchten Wortes eingeben. Nach der Eingabe von CR erscheint der Bildschirm mit dem Text in Arbeit wieder. Der Cursor blinkt nun hinter dem ersten Wort, das dem Gesuchten entspricht. Wenn man anschließend die Tasten CTRL-L drückt, so wird das nächste entsprechende Wort gesucht. Wenn kein Wort gefunden wurde, erscheint der Cursor ganz am Ende des Textes.

Wie man Wörter ersetzt

Man kann auch nach einem Wort suchen, um es zu ersetzen. Dazu gibt man nach CTRL-Q die Tasten SHIFT-A ein. Dann erscheint wieder das Feld für die Eingabe einer Zeichenfolge, die gesucht werden soll. Nach CR erscheint jetzt aber erst ein Feld für eine Zeichenfolge, die anstelle der gesuchten einzusetzen ist. Wenn man nun abschließend CR eingibt, so beginnt die Suche. Falls ein Wort gefunden wurde, so blinkt der Cursor diesmal am Anfang des Wortes. Wenn man dieses Wort durch das neu angegebene ersetzen will, so muß man die Ta-

ste J betätigen. Dann wird die alte Zeichenfolge gelöscht und die neue eingefügt. Anschließend wird weitergesucht. Wenn man die Taste N betätigt, so bleibt das gefundene Wort bestehen, die Suche wird ohne Änderung fortgeführt. Wenn man dabei eine andere Taste als J oder N eingibt, so wird die Suche abgebrochen.

Man kann sie jederzeit mit CTRL-L wieder neu aufnehmen, ohne die Zeichenfolgen neu eingeben zu müssen.
Eine weitere wichtige Umschaltfunktion besitzt der Befehl CTRL-K. Wenn man danach die Tasten SHIFT und X drückt, so wird der Editor korrekt verlassen und man gelangt zurück ins Grundmenü. Man darf den Editor übrigens nie durch Drücken der Reset-Taste verlassen, denn dabei geht der Text verloren, der zuletzt im Bildfenster des Editors sichtbar war.

Blöcke bilden

Mit der Umschaltfunktion CTRL-K kann man als Besitzer der neuen Grundprogrammversion 4.2 jetzt auch Blöcke bearbeiten (alle Steuerfunktionen zeigen Bild 22 und 23).

Der Block, der an beliebiger Stelle des Textes steht, wird zunächst markiert. Mit dem Befehl CTRL-K und B wird der Blockanfang durch einen kleinen Pfeil gekennzeichnet. Mit CTRL-K und K wird das Blockende gekennzeichnet, z.B. wie in Bild 22.

Damit die Blockbefehle arbeiten, muß das Blockende stets hinter dem Blockanfang stehen. Nun kann man den Block kopieren mit CTRL-K und C (Bild 23), oder ihn transportieren mit CTRL-K und V, oder löschen mit CTRL-K und Y.

Wenn man CTRL-K und W eingibt, so kann man den Block in einen beliebigen Speicherbereich transportieren. Dazu wird nach der Zieladresse (*Bild 24*) gefragt. Man gebe hier eine beliebige Adresse ein, auf der sich RAM befindet (in unserem Beispiel $9800, das Dollarzeichen gibt an, daß es sich um eine sedezimale Zahl handelt).

Mit CTRL-K und R kann man Texte aus einem anderen Speicherbereich in das Textfenster einfügen. Der Text wird dabei, beginnend bei der aktuellen Cursorstelle, eingesetzt. Dazu benötigt man die Blockmarken nicht.

Die Blockmarken werden automatisch bei Verlassen des Editors entfernt. Man kann sie jedoch auch von Hand (durch Eingabe von CTRL-K und H) entfernen lassen. Bevor man neue Marken setzt, sollte man dies immer tun, um die Bearbeitung übersichtlicher zu gestalten.
Nun noch eine Besonderheit. Wenn man mehr als 40 Zeichen in eine Zeile tippt, so verschwindet der Cursor aus dem Bildfeld. Die Zeichen werden aber trotzdem abgelegt. Man kann den Bildschirm für solche Fälle auf 80 Zeichen pro Zeile umschalten. Dazu gehe man in das Menü OPTIONEN und dort in das Menü „4 = Zeichen/Zeile & Debug". Dann wählt man „2 = 80 Zeichen/Zeile" und geht zurück in den Editor. Man erhält dann *Bild 25*. Dort gehen jetzt 80 Zeichen in eine Zeile, weil der Text enger steht.

Das Textfenster

Vielleicht haben Sie sich schon Gedanken über die Informationen in der untersten Bildschirmzeile gemacht, die im Editorfenster (z.B. *Bild 25*) zu sehen ist. Dort steht „Textstart = 009000" und anderes mehr.

Dazu zeigt *Bild 26* eine Speicheraufteilung. Der Speicher beginnt normalerweise bei Adresse $8000 (sedezimal) und endet, wenn man nur einen RAM-Baustein eingebaut hat, bei $9FFF. Den Bereich $8000 bis $8FFF sollte man nicht verwenden, er ist für das Grundprogramm reserviert. Ab $9000 beginnt der Textspeicher, die Adresse $9000 ist nach dem Einschalten des Grundprogramms vorgegeben, sie kann jedoch beliebig geändert werden, wenn man im Menü OPTIONEN mit „2 = Textstart" eine neue Adresse angibt. Damit kann man auch mehrere unterschiedliche Textgebiete im Speicher halten.

Im Editor ist zur Kontrolle der jeweils aktuelle Textanfang als Adresse in Editorfenster ausgegeben. Die Ausgabe „Fenster=" (in der alten Revision stand hier „Cursor=") ist die Adresse, bei der der erste Buchstabe im Textfenster beginnt. Alles, was im Textfenster sichtbar ist, befindet sich zu diesem Zeitpunkt nicht im Textgebiet, sondern in einem getrennten Speicherbereich ($8000 bis $8FFF). Die letzte Angabe „Tor=" gibt das aktuelle Textende an, das jedoch nicht das endgültige Textende darstellt, denn der Text im Textfenster wird zunächst nicht mitgerechnet. Um festzustellen, wie lang ein Text ist, muß man mit CTRL-Q und C und dann CTRL-C so weit ans Textende gehen, daß kein Text mehr im Textfenster sichtbar ist, also nur noch die Feldzeichen (schraffierte

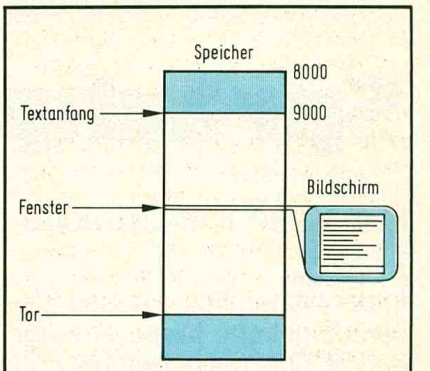


Bild 26. Die Aufteilung des Speichers für den Editor

Zeichen ganz links) erscheinen. Dann erst gibt die Adresse „Tor=" das richtige Textende an. Diese Informationen sind später wichtig, wenn man große Programme eingeben will, und der Text Gefahr läuft, das Programm zu überschreiben. Dann muß man das Programm verschieben, um Platz zu gewinnen. Doch das wird später noch genau erläutert.

Aufgaben

1. Welche Aufgabe hat der Baustein 555 in der CPU-Baugruppe?

2. Welche Periodendauer in ns (Nanosekunden) besitzt ein 8-MHz-Takt-Signal?

3. Was geschieht, wenn man Wartezyklen an der CPU-Baugruppe einstellt?
4. Auf der ROA64 seien die Brücken 16 und 18 eingesetzt, die Brücken 17 und 19 offen. Auf welche Adresse spricht die Karte an? (Man kann das überprüfen, wenn man sich eine zweite ROA-Karte leiht und diese mit einem RAM-Baustein bestückt und mit den Brücken einbaut, dann wählt man den Menüpunkt „Speicherbereiche" und erhält die Anfangsadresse der Baugruppe.)

5. Was muß ein Texteditor mindestens können, damit man sinnvoll arbeiten kann?

Rolf-Dieter Klein

Sprachprobleme

Die Maschinensprache des 68008

In der letzten Folge haben Sie die CPU 68008 und den Texteditor kennengelernt. In dieser Folge werden einige Befehle des 68008 erklärt. Daß diese Befehle anders aussehen als beim Z80, ist klar. Dennoch gibt es Gemeinsamkeiten. An Beispielen sollen Sie die 68008-Befehle nach und nach kennenlernen.

Um schnell zu Ergebnissen zu kommen, seien wieder Elemente der grafischen Sprache verwendet, die auch schon beim Z80 als Lernhilfe gedient hat. Die erste Aufgabe besteht darin, ein Quadrat zu zeichnen. *Bild 1* zeigt die Lösung, einmal im 68008-Assembler geschrieben und einmal im Z80-Maschinencode.

Zunächst fällt einmal auf, daß das 68008-Programm länger ist. Das liegt daran, das hier noch nicht alle Möglichkeiten ausgeschöpft sind und zum anderen, daß alle Befehle beim 68008 nicht im Maschinencode, sondern in Assembler geschrieben sind. In der Assembler-Sprache hat jeder Befehl einen Namen. In einem Assemblerprogramm muß man diese Namen benutzen. Der Assemblertext wird später von einem Programm, Assembler genannt, in den Maschinencode übersetzt. Damit spart man sich die Mühe, in einer beim 68008 sehr umfangreichen Tabelle nachzuschlagen und kann in einigermaßen verständlichen Wörtern programmieren.

Quadrat

In der ersten Zeile des Assemblerprogramms steht die Anweisung „ORG $9C00". Damit wird dem übersetzenden Assembler gesagt, wohin er den erzeugten Maschinencode legen soll. Das Zeichen Dollar gibt dabei an, daß die nachfolgende Zahl im sedezimalen Zahlensystem dargestellt wird. Das übersetzte Programm wird also ab Adresse 9C00 abgelegt. Dann folgen zwei Anweisungen mit EQU. Dabei wird dem links von EQU stehenden Symbol zu Beginn der Assemblierung ein Wert zugeordnet. So werden die beiden Symbole SCHREITE und DREHE definiert.

68008-Assemblerprogramm

```
   org $9c00
schreite equ 1
drehe    equ 2
quadrat:
   move.w #4-1,d3
schleife:
   move.w #50,d0
   move.w #schreite,d7
   trap #1
   move.w #90,d0
   move.w #drehe,d7
   trap #1
   dbra d3,schleife
   rts

   end
```

Z80-Programm

```
QUADRAT:=$
 21 4.W
 CD SCHLEIFE
 21 #50.W
 CD SCHREITE

 21 #90.W
 CD DREHE

 CD ENDSCHLEIFE
 C9
```

Bild 1. Zwei Programme für verschiedene Prozessoren mit identischer Wirkung auf den Bildschirm

Im eigentlichen Programmblock wird als erstes wieder ein Symbol definiert. Diesmal handelt es sich um eine sogenannte Marke, die durch das Doppelpunkt-Zeichen definiert wird. Das entspricht der ersten Zeile des Z80-Zeichensprachen-Programms. Die Marke QUADRAT erhält hier als Wert die aktuelle Adresse des Maschinenspracheprogramms, sobald der Assembler beim Übersetzen den Doppelpunkt erkennt. Dieser aktuelle Wert lautet wiederum 9C00, weil es die erste Zeile ist.

MOVE und TRAP

Jetzt folgt die erste 68008-spezifische Anweisung, der MOVE-Befehl. MOVE kommt aus dem Englischen und bedeutet soviel wie „bewege". Bewegt wird der Wert 4–1. Aus diesem Ausdruck macht der Assembler automatisch eine Drei, die dann beim Programmlauf in D3 gegeben wird. Das Zeichen # besagt, daß der Wert direkt verwendet wird, also keine Adresse bezeichnet. Man hätte auch #3 schreiben können, jedoch soll hier daran erinnert werden, daß man vier Schleifendurchläufe benötigt. Man muß in solchen Fällen aus technischen Gründen beim 68008 manchmal einen Wert laden, der um eins verringert ist. Im Programmtext kommt jetzt eine weitere Marke, mit dem Namen SCHLEIFE. Hier beginnt eine Wiederholungsschleife, die viermal durchlaufen werden soll. Deren erster Befehl lädt mit MOVE den Wert 50 in das Register D0. Übrigens, das .W gibt an, daß eine 16-Bit-Größe bewegt werden soll. *Bild 2* zeigt eine Zusammenstellung der möglichen Attribute für Größen.

.B	Byte	8	Bit
.W	Wort	16	Bit
.L	Langwort	32	Bit

Bild 2. Das sind die Attribute für Größen, die der Assembler versteht

Im zweiten Befehl innerhalb der Schleife wird der Wert des Symbols SCHREITE in das Register D7 geladen. Der Wert von SCHREITE ist 1, man hätte also auch MOVE.W #1,D7 schreiben können, aber die Verwendung von Symbolen macht Programme meist übersichtlicher. Nun folgt der Befehl TRAP #1. Er ist eine Besonderheit beim 68008. Er ist eine Art von Unterprogrammaufruf. Die Adresse des Unterprogramms wird durch das Grundprogramm festgelegt. Der Parame-

ter des TRAP-Befehls kann von 0 bis 15 reichen, entsprechend 16 verschiedenen Einsprüngen. Wir benötigen hier TRAP#1.
Damit sind eine Vielzahl von Unterprogrammen erreichbar, die im Grundprogramm stehen. Dazu wird dem TRAP-Befehl in Register D7 der sogenannte Index der Befehlstabelle mitgegeben. Der erste Eintrag in der Befehlstabelle ist der Schreite-Befehl, der zweite der Drehe-Befehl usw.

Durch den TRAP-Befehl wird also zunächst der Schreite-Befehl angesteuert. Und der Schreitebefehl verlangt als Parameter einen Zahlenwert in Register D0. Es werden 16 Bits benötigt, weshalb beim vorhergehenden MOVE-Befehl „.W" als Attribut angegeben wurde. Die nächsten Befehle laden den 16-Bit-Wert 90 ins Register D0 und den Wert für DREHE ins Register D7. DREHE = 2 dient als Index für den erneut folgenden TRAP-Befehl, der diesmal eben den Befehl DREHE ansteuern soll. Insgesamt wurde also eine Linie gezeichnet und die Schildkröte anschließend um 90 Grad gedreht. Nun folgt der Schleifen-Wiederhol-Befehl.

Decrement und Branch

Dazu gibt es beim 68008 eine sehr raffinierte Anweisung, DBRA (Decrement und Branch), was soviel heißt wie verringere und springe. Dazu muß man ein Register angeben, dessen Inhalt verringert werden soll. Hier sind nur Datenregister erlaubt. Hinter einem Komma steht dann die Adresse des Sprungziels, das hier SCHLEIFE lautet. Der DBRA-Befehl verringert das Datenregister D3 zunächst um den Wert 1. Dann wird geprüft, ob der neue Wert –1 ist. Wenn nein, so wird zur angegebenen Marke gesprungen, hier die Schleife also wiederholt. Wenn ja, so wird der auf DBRA folgende Befehl ausgeführt, also die Schleife verlassen.

RTS

Der nachfolgende Befehl lautet RTS, was bedeutet RETURN FROM SUBROUTINE, auf deutsch Unterprogramm-Rückkehr.
Weshalb wird die Schleife nun viermal durchlaufen? Dazu betrachte man das Register D3. Vor Beginn des Schleifendurchlaufs wird es mit dem Wert 3 geladen. Beim ersten Durchlauf wird es beim DBRA-Befehl um eins verringert, also auf den Wert 2 gebracht. Dann beim nächsten Durchlauf auf 1, dann auf 0 und dann auf –1. Erst beim vierten Mal enthält D3 den Wert –1 und die Schleife wird beendet.

So wird das Programm eingegeben

Mit dem Texteditor haben Sie ja schon kurz gearbeitet. Dieser wird nun als erstes vom Menü aus aufgerufen (1=EDITOR). Unter dem Editor tippt man das Programm ein, und erhält *Bild 3*, wenn das gelungen ist. Man achte peinlich auf alle Punkte und Kommas, die abgedruckt sind, denn der Computer versteht nur Programme ohne Tippfehler. Dabei sind ihm aber zusätzliche Leertasten zwischen Wörtern in einer Zeile egal.
Nun wird der Editor durch Eingabe von CTRL-K und Shift-X verlassen. Aus dem jetzt auf dem Bildschirm erscheinenden Menü wird nun ‚2=Assembler' aufgerufen. Das Programm wird dadurch automatisch in Maschinencode übersetzt. Am Schluß ergibt sich *Bild 4*.
Bei älteren Grundprogrammen zeigt sich unten noch die Bemerkung „000000 Fehler", also keine Fehler. Wenn Sie einen Tippfehler gemacht haben, so markiert der Assembler die Stelle durch einen Pfeil. Dann muß man wieder in den Editor zurück und den Fehler korrigieren. Danach kann ein neuer Übersetzungsversuch gestartet werden. Diese Prozedur muß man so lange wiederholen, bis keine Fehler mehr vom Assembler reklamiert werden.
Ist das der Fall, dann kann man das Programm starten. Dazu geht man mit M ins Haupt-Menü zurück und wählt die Funktion 2 = starten auf. Es erscheint *Bild 5*. Dann wird der Name des Programms angegeben, also hier der Name QUADRAT, wie er im Assemblertext definiert wurde. Nach Eingabe von CR (Carriage Return) erscheint *Bild 6* und man kann die Schildkröte durch Eingabe des Buchstabens F ausblenden, um ein flimmerfreies Bild zu erhalten.
Die Definition der Symbole SCHREITE und DREHE war mit EQU schnell getan.

```
▒
▒

* Quadratprogramm

 org $9c00
schreite equ 1
drehe equ 2

quadrat:
 move.w #4-1,d3
schleife:
 move.w #50,d0
 move.w #schreite,d7
 trap #1
 move.w #90,d0
 move.w #drehe,d7
 trap #1
 dbra d3,schleife
 rts

 end
▒
Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

Bild 3. So sieht das Quadratprogramm beim Editieren aus

```
009C00
009C00                          * QUADRATPROGRAMM
009C00
009C00                           ORG $9C00
= 00000001                      SCHREITE EQU 1
= 00000002                      DREHE EQU 2
009C00
009C00                          QUADRAT:
009C00  363C 0003                MOVE.W #4-1,D3
009C04                          SCHLEIFE:
009C04  303C 0032                MOVE.W #50,D0
009C08  3E3C 0001                MOVE.W #SCHREITE,D
009C0C  4E41                     TRAP #1
009C0E  303C 005A                MOVE.W #90,D0
009C12  3E3C 0002                MOVE.W #DREHE,D7
009C16  4E41                     TRAP #1
009C18  51CB FFEA                DBRA D3,SCHLEIFE
009C1C  4E75                     RTS
009C1E
009C1E                           END

008B4A  Ende-Symboltabelle
```

Bild 4. Das macht der Assembler daraus

Bild 5. Quadrat wird gestartet

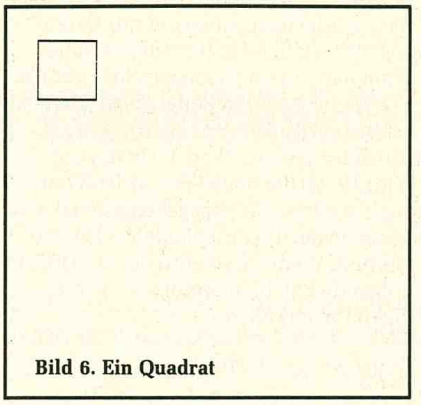


Bild 6. Ein Quadrat

Wenn man sich alle Symbole des Grundprogramms stets auf diese Weise definieren müßte, wäre das schwierig. Das Grundprogramm kennt aber alle seine Symbole selbst (bei Version 4.0 sind das 120 verschiedene). Man kann sie im Assemblertext direkt verwenden. Dazu muß man sie, wie in *Bild 7* gezeigt, mit einem Ausrufezeichen eingeben. Durch das Ausrufezeichen wird dem Assembler mitgeteilt, daß ein fest eingebautes Symbol folgt. Es wird daraufhin der Index berechnet, wie ihn der TRAP#1-Befehl benötigt. Das Programm in Bild 7 ist kürzer als das erste. Eine weitere Besonderheit ist hier, daß der ORG-Befehl fehlt. Wenn man keinen ORG-Befehl verwendet, so läßt der Assembler das Maschinenprogramm stets automatisch bei Adresse $9C00 beginnen. Man muß ORG nur verwenden, wenn man das Programm an einer anderen Stelle ablegen will.

Die Speicherbereiche

Ein kleines Diagramm dazu zeigt *Bild 8*. Es gibt vier Speicherbereiche, deren Ausdehnung sich ständig ändern kann, die das Grundprogramm verwaltet. Zum einen müssen Symbole im Speicher abgelegt werden. Das Grundprogramm legt sie direkt über den eigenen Variablen in aufsteigender Reihenfolge ab. Dann beginnt das Textgebiet für Programmtexte ab Adresse $9000 – wenn man es nicht (Optionen-Menü) anders bestimmt hat. Auch dieses Gebiet ist unterschiedlich lang, je nachdem, wieviel Text man eingegeben hat. Darüber liegt das Gebiet für Maschinencode. Es beginnt ab Adresse $9C00 – wenn man keine andere ORG-Anweisung gegeben hat. Ein Programm wächst im Speicher nach oben hin. Wenn sich die genannten Gebiete durch Wachstum während der Arbeit überlappen, so gibt es Fehler, die im schlimmsten Fall zum Verlust des Programms führen können. Daher sollten Sie nach einer Texteingbe zunächst den Text unter Menü CAS auf Kassette speichern und prüfen. Dann erst sollten Sie den Assembler starten, denn die Symboltabelle beginnt jetzt zu wachsen und könnte den Textanfang erreichen. Bei der Assemblerausgabe erscheint im unteren Bild immer die Angabe der zuletzt verwendeten Adresse für Symbole. Wenn diese zu nahe an den Programmtextanfang herankommt, muß man den Text weiter nach oben verschieben (entweder mit dem Blockwritebefehl oder indem man das Programm von Kassette neu lädt und eine höhere Textadresse angibt). Wenn der Programmtext zu lange wird, und auf den Maschinencodebereich zusteuert („TOR =“ oder „Textende =“ bei älteren Versionen beachten!), so muß man mit der ORG-Anweisung arbeiten und eine Adresse angeben, die garantiert noch nicht vom Text belegt ist. Achtung, die angebene Adresse im Editor ist nur dann die Endadresse, wenn kein Text mehr im Bildfenster sichtbar ist (also mit CTRL-Q und C, sowie CTRL-C den Text herausschieben). Bei unseren ersten kleinen Programmen muß man sich um solche Probleme aber noch nicht kümmern.

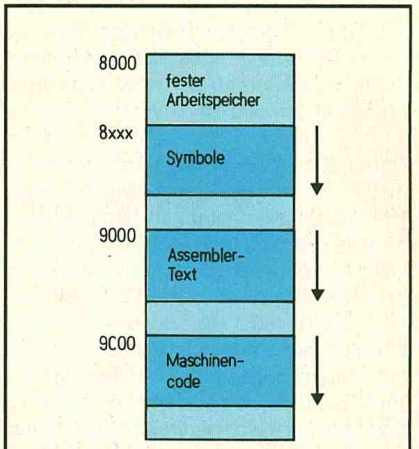


Bild 8. Die Speicheraufteilung unter dem Grundprogramm

Der JSR-Befehl

Nun noch eine weitere Möglichkeit der Symbolangabe. Dazu wird ein anderer Unterprogrammaufruf verwendet, der

```
*
* Quadratprogramm

quadrat:
 move.w #4-1,d3
schleife:
 move.w #50,d0
 move.w #!schreite,d7
 trap #1
 move.w #90,d0
 move.w #!drehe,d7
 trap #1
 dbra d3,schleife
 rts

 end
```

Textstart=009000 Fenster=009000 Tor=009000 amer CTRL-J=Hilfe

Bild 7. Ein neues Quadratprogramm

JSR-Befehl. Er erhält als Parameter die absolute Adresse eines Unterprogramms. *Bild 9* zeigt das Programm in dieser neuen Form. Das Zeichen @ gibt dem Assembler an, daß ein Symbol für eine absolute Adresse verwendet werden soll. Dann wird automatisch die zugehörige Adresse während der Übersetzung eingesetzt. Achtung! Diese Programmform hat den Nachteil, daß nun absolute Adressen im übersetzten Programm vorkommen. Wenn man das Grundprogramm ändert, z. B. gegen eine neuere Version austauscht, so muß man dann alle Programme neu übersetzen, um jeweils die veränderten Adressen in die Programme einzutragen.
Bild 10 zeigt die dazugehörige Assemblerausgabe, diesmal mit 80 Zeichen pro Zeile. Wenn man größere Programme schreibt, so ist es ganz nützlich, wenn man die Assemblerausgabe ganz ausschalten kann (im Optionen-Menü die Option 1 = Assembler-Optionen auswählen und dort 1 = nur Fehlerausgabe). Dann wird die Übersetzung sehr rasch durchgeführt.

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

009C00
009C00                  * QUADRATPROGRAMM
009C00
009C00                  QUADRAT:
009C00  363C 0003        MOVE.W #4-1,D3
009C04                  SCHLEIFE:
009C04  303C 0032        MOVE.W #50,D0
009C08  4EB9 00001288    JSR @SCHREITE
009C0E  303C 005A        MOVE.W #90,D0
009C12  4EB9 000012F6    JSR @DREHE
009C18  51CB FFEA        DBRA D3,SCHLEIFE
009C1C  4E75             RTS
009C1E
009C1E                  END

008B4A  Ende-Symboltabelle
```

Bild 10. Das Programm in 68008-Maschinensprache

Fehler, Fehler!

Einmal gibt es beim Programmieren Fehler, die schon der Assembler erkennt. Falsch geschriebene Namen z. B. oder fehlende Kommata, wie auch undefinierte Symbole. Dann gibt es noch logische Fehler, die man erst nach einem Programmstart bemerken kann. Das sind Fehler, die auch als Denkfehler bezeichnet werden könnten. Um diese Fehler aufzuspüren und zu eliminieren, gibt es ein Hilfsmittel, den Einzelschritt im Grundprogramm. Zunächst wird dazu das Quellprogramm mit Kommentaren versehen, wie es *Bild 11* zeigt. Alle Text in einer Zeile hinter dem Zeichen * werden nämlich vom Assembler ignoriert. Sie können also dazu verwendet werden, Programmstellen zu erläutern. Das ist für das spätere Verständnis wichtig. Nun geht man aus dem Editor heraus und wählt das Optionen-Menü an. Dort wählt man die Funktion 4 = Zeichen/Zeile & Debug. Danach die Funktion 3 = Debug Info an. Danach startet man den Assembler und man erhält *Bild 12*. Es wird hier zusätzlich die Endadresse der Debugtabelle angegeben, die direkt hinter dem Quelltext steht. In der Debugtabelle befindet sich eine Zusatzinformation, die beim Einzelschritt benötigt wird. Das Einzelschritt-Menü wird nun aufgerufen und die Startadresse wie in *Bild 13* eingegeben. Dann erscheint *Bild 14*. Unten rechts erscheint die Quellzeile eingeblendet, die als erstes aufgeführt wird. Im restlichen Feld unten sieht man den aktuellen Inhalt aller Register des 68008.
Durch Drücken der Taste CR kann man nun das Programm Schritt für Schritt ausführen. *Bild 15* zeigt ein fortgeschrittenes Stadium. Nun versteht man auch, warum die Kommentare eingefügt wurden. Da sie auch im Bild erscheinen, kann man sich im Programm leichter zurechtfinden. Die Debugtabelle, die der Assembler dazu erstellt, enthält jeweils ein Wertepaar pro Befehlszeile. Der erste Wert ist die Adresse des Programmzählers, bei dem der Befehl auftritt, und der zweite Wert ist die Adresse der Quellzeile. Damit ist eine eindeutige Zuordnung Programmzeile ↔ Kommentar möglich.

MOVE: Der Inhalt eines Quellgebietes wird in ein Zielgebiet bewegt

Bild 16 zeigt den MOVE-Befehl. Der MOVE-Befehl ist sehr universell, was die Angabe von Quell- und Zieladresse angeht. Wir haben zunächst nur ein paar einfache Möglichkeiten kennengelernt. Grundsätzlich kann man sowohl Register als auch Speicherstellen im MOVE-

```
* Quadratprogramm

quadrat:
 move.w #4-1,d3
schleife:
 move.w #50,d0
 jsr @schreite
 move.w #90,d0
 jsr @drehe
 dbra d3,schleife
 rts

 end

Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

Bild 9. Nochmals ein Quadrat

```
* Quadratprogramm

quadrat:
 move.w #4-1,d3       * vier Mal
schleife:
 move.w #50,d0        * Seitenlaenge
 jsr @schreite        * Vorwaerts
 move.w #90,d0        * Winkel
 jsr @drehe           * Drehen
 dbra d3,schleife     * wieder zuerueck
 rts                  * Ende

 end

Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

Bild 11. Kommentare sind wichtig

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 19E


009C00
009C00                          * QUADRATPROGRAMM
009C00
009C00                          QUADRAT:
009C00   363C 0003               MOVE.W #4-1,D3
009C04                          SCHLEIFE:
009C04   303C 0032               MOVE.W #50,D0
009C08   4EB9 00001288           JSR @SCHREITE
009C0E   303C 005A               MOVE.W #90,D0
009C12   4EB9 000012F6           JSR @DREHE
009C18   51CB FFEA               DBRA D3,SCHLEIFE
009C1C   4E75                    RTS
009C1E
009C1E                          END

008B4A   Ende-Symboltabelle
009166   Ende-Debug-Tabelle


                                                F=Flip M=Menue
```

Bild 12. Der Assembler gibt dieses Bild aus, wenn er mit „Debug Info an" gestartet wird

```
D0       D1       D2       D3       D4       D5       D6       D7
00010001 00010080 00015000 0000FFFF 00015000 FFFF0000 0000A830 000005D0
A0       A1       A2       A3       A4       A5       A6       A7
000033D4 00008000 00000480 FFFF7FFF 0001A052 00008000 00080A00 0001FFF0
SR   USP      SSP      PC
A700 0001FFF0 0001FFF0 00009C00  move.w #4-1,d3     * vier Mal
```

Bild 14. Das zeigt sich nach dem Start: Die Registerinhalte und der auszuführende Befehl stehen da

```
D0       D1       D2       D3       D4       D5       D6       D7
00000000 000100CE 00010099 00000001 00015000 FFFF0000 0000A830 000005D0
A0       A1       A2       A3       A4       A5       A6       A7
000033D4 00008000 00000480 FFFF7FFF 0001A052 00008000 00080A00 0001FFF0
SR   USP      SSP      PC
A704 0001FFF0 0001FFF0 00009C04  move.w #50,d0      * Seitenlaenge
```

Bild 15. Schritt für Schritt wird das Quadrat gezeichnet

Befehl als Quelle und Ziel angeben. Direkte Werte lassen sich aber nur im Quellen-Feld angeben. *Bild 17* zeigt, wie der Assembler Adressen von Werten unterscheidet. Bei der direkten Wertangabe wird das Zeichen # dem Wort vorangestellt. Eine Adreßangabe enthält dieses Zeichen nicht. Werte wiederum kann man unterschiedlich angeben. Einmal als Zahlenwert, zum anderen als Symbol. Zusätzlich kann man unterschiedliche Zahlensysteme verwenden oder sogar Rechnungen vom Assembler durchführen lassen, wie 4–1. *Bild 18* zeigt ein paar Beispiele für diese Techniken.
In *Bild 19* sind alle Fähigkeiten aufgezeigt, die der Assembler während der Übersetzung anwenden kann. Übrigens können diese Techniken auch im Grundprogramm verwendet werden, immer wenn ein Eingabefeld für Werte, wie Startadresse usw., erscheint. Soll zum Beispiel der ASCII-Code des Buchstabens „A" in das Register D0 geladen werden, so verwendet man den Befehl MOVE.B#'A',D0. Durch B wird angegeben, daß man nur ein Byte laden möchte. Alle Assemblerfunktionen mit dem Hinweis 4.0 sind erst ab der Grundprogrammversion 4.0 verfügbar. Die Multiplikation und Division werden erst im Grundprogramm 4.0 mit 32 Bit ausgeführt, früher wurden nur 16 Bit verknüpft, was in den meisten Fällen jedoch ausreicht. Es muß betont werden, daß diese Operationen alle nur den Assembler betreffen und nicht im laufenden Programm wirksam werden.
Ein paar Funktionen auslösende Zeichen kommen doppelt vor, so das Ausrufezeichen oder der Stern. Der Assem-

Bild 13. Start im Einzelschritt

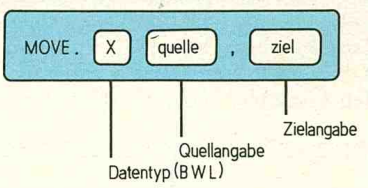


Bild 16. Der MOVE-Befehl und seine Parameter

Bild 17. Das Doppelkreuz unterscheidet direkte Werte von Adreßangaben

100	dezimal
$AFFF	sedezimal
%10100101	binär
100*3+5-4	Ausdruck
name	symbolisch
name+$AF	gemischt

Bild 18. Größen können sehr verschieden eingebracht werden

+	Addition 32 Bit
-	Subtraktion 32 Bit
*	Multiplikation 32 Bit ab 4.0
/	Division 32 Bit ab 4.0
\	Modulodivision ab 4.0
( )	Klammerrechnung
!	Oder-Verknüpfung ab 4.0
&	Und-Verknüpfung ab 4.0
*	Als Operand, Programmstand
?	Textanfangsadresse
'text'	Text, als ASCII-Wert
~	Nicht-Verknüpfung
-	Vorzeichen einer Zahl
$nnnnnnnn	sedezimale Zahl
ddddddddd	dezimale Zahl
%bbbbbbbb	binäre Zahl
@symbol	Adresse eines festen Symbols
!symbol	Index eines festen Symbols

Bild 19. Das sind die arithmetischen und logischen Operationen, die der Assembler beim Assemblieren durchführen kann

bler erkennt deren Bedeutung anhand des Zusammenhangs immer eindeutig. So bedeutet der Ausdruck *+2 eine Adresse, die sich aus der aktuellen Programmzähleradresse plus dem Wert 2 errechnet. Der Programmzählerstand ist dabei immer die Speicher-Adresse, bei der der Assembler den Maschinenbefehl ablegt, dessen Adreßteil als solch ein Ausdruck im Assemblerprogramm angegeben ist. Der Maschinencode für einen Befehl kann übrigens, je nach Befehl, aus aufeinanderfolgenden Bytes unterschiedlicher Anzahl bestehen.

```
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

009C00                          START:
009C00  4EB9 00001112            JSR @HIDE
009C06  3C3C 0009                MOVE #10-1,D6
009C0A                          ST11:
009C0A  4EB9 00009C26            JSR MEHR
009C10  303C 0024                MOVE #36,D0
009C14  4EB9 000012F6            JSR @DREHE
009C1A  51CE FFEE                DBRA D6,ST11
009C1E  4EB9 00000A00            JSR @CI
009C24  4E75                     RTS
009C26
009C26
009C26                          MEHR:
009C26  3A3C 0009                MOVE #10-1,D5
009C2A                          ST1:
009C2A  4EB9 00009C40            JSR BLUME
009C30  303C 0024                MOVE #36,D0
009C34  4EB9 000012F6            JSR @DREHE
009C3A  51CD FFEE                DBRA D5,ST1
009C3E  4E75                     RTS
009C40
009C40
009C40                          BLUME:
009C40  6100 0044                BSR BLUETE
009C44  303C FFB0                MOVE #-80,D0
009C48  4EB9 00001288            JSR @SCHREITE
009C4E  4E75                     RTS
009C50
009C50  4EB9 00009CA0            JSR BLATT
009C56  303C FFFB                MOVE #-5,D0
009C5A  4EB9 00001288            JSR @SCHREITE
009C60  303C FFA6                MOVE #-90,D0
009C64  4EB9 000012F6            JSR @DREHE
009C6A  4EB9 00009CA0            JSR BLATT
009C70  303C 005A                MOVE #90,D0
009C74  4EB9 000012F6            JSR @DREHE
009C7A  303C FFEC                MOVE #-20,D0
009C7E  4EB9 00001288            JSR @SCHREITE
009C84  4E75                     RTS
009C86
009C86
009C86                          BLUETE:
009C86  363C 0004                MOVE #5-1,D3
009C8A                          BL1:
009C8A  4EB9 00009CA0            JSR BLATT
009C90  303C 0048                MOVE #360/5,D0
009C94  4EB9 000012F6            JSR @DREHE
009C9A  51CB FFEE                DBRA D3,BL1
009C9E  4E75                     RTS
009CA0
009CA0
009CA0                          BLATT:
009CA0  4EB9 00009CC2            JSR VIERTEL
009CA6  303C 005A                MOVE #90,D0
009CAA  4EB9 000012F6            JSR @DREHE
009CB0  4EB9 00009CC2            JSR VIERTEL
009CB6  303C 005A                MOVE #90,D0
009CBA  4EB9 000012F6            JSR @DREHE
009CC0  4E75                     RTS
009CC2
009CC2
009CC2
009CC2                          VIERTEL:
009CC2  383C 002C                MOVE #45-1,D4
009CC6                          LOOP:
009CC6  303C 0002                MOVE #2,D0
009CCA  4EB9 000012F6            JSR @DREHE
009CD0  303C 000A                MOVE #10,D0
009CD4  4EB9 0000128A            JSR @SCHR16TEL
009CDA  51CC FFEA                DBRA D4,LOOP
009CDE  4E75                     RTS
009CE0
009CE0
009CE0

008B02  Ende-Symboltabelle
```

Bild 20. Das Programm . . .

Ein längeres Programm

In *Bild 20* ist ein umfangreicheres Programm abgebildet. Bei manchen Druckern wird übrigens das Zeichen „@“ durch „§“ dargestellt. Solche Problemchen gibt es oft in der Mikro-Computerei. Dann gibt es noch den Aufruf JSR @SCHR16TEL. Damit ist es möglich 1/16-Schritte auszuführen, um auch kleine Figuren darstellen zu können. *Bild 21* zeigt das Ergebnis eines Programmlaufs. *Bild 22* zeigt eine Zusammenstellung der wichtigsten Schildkrötenbefehle aus der Grafiksprache, wie sie für die Darstellung von unterschiedlichen Figuren benötigt werden. Mit diesen Befehlen kann man komplexe Grafiken erzeugen, ohne weitere 68008-Befehle verwenden zu müssen.
SCHREITE und DREHE sind ja schon bekannt. Mit HEBE wird erreicht, daß keine Schreibspur mehr hinterlassen wird und mit SENKE wird die Schreibspur wieder eingeschaltet. Mit SET kann man die Schildkröte auf eine absolute Position stellen und SCHR16TEL funktioniert wie SCHREITE, nur mit 1/16 Schrittweite. Mit HIDE kann man die Seitenumschaltung unterdrücken und dadurch die Schildkröte ausblenden. Mit SHOW wird sie wieder eingeschaltet.

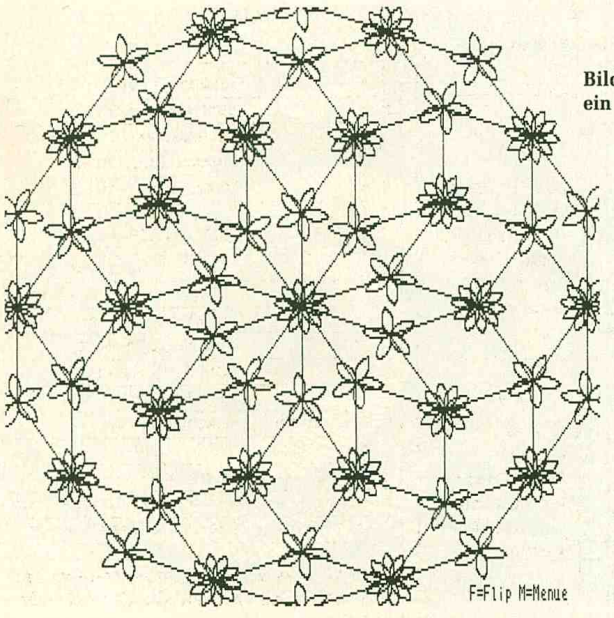


Bild 21. . . . zeichnet ein Blumengebinde

BRA

In *Bild 23* wird ein neuer Befehl verwendet. Der Befehl BRA. Damit wird ein sogenannter unbedingter Sprung ausgeführt. Die Ausführung des Programms

1	SCHREITE	D0.W=Anzahl
2	DREHE	D0.W=Winkel
3	HEBE	
4	SENKE	
5	SET	D1.W=X,D2.W=Y, D3.W=Winkel
19	SCHR16TEL	D0.W=Anzahl
47	HIDE	
48	SHOW	

Bild 22. Die Befehle zur Schildkrötengrafik im 68008-Grundprogramm

wird an der nach BRA durch das Symbol NEWPOLY angegebenen Adresse fortgeführt. Damit lassen sich zum Beispiel Endlosschleifen realisieren. Will man ein solches Programm übrigens anhalten, so muß man die RESET-Taste drükken. Programme und Quelltexte bleiben bei Reset erhalten, so daß kein Programmverlust zu befürchten ist.
Eine weitere Besonderheit ist, daß bei den MOVE-Befehlen in Bild 23 die Ergänzung „W" fehlt. Immer, wenn man jede Ergänzung wegläßt, so ist bei unserem 68008-Assembler automatisch eine Wortgröße gemeint. Erst wenn man ein Langwort oder ein Byte meint, muß man eine Ergänzung (.L oder .B) angeben. *Bild 24* zeigt die Ausgabe auf dem Bildschirm, die durch das Programm in Bild 23 erzeugt wird. Man kann durch Ändern der EQU-Anweisung weitere Figuren erzeugen. Eigene Experimente seien empfohlen.

```
winkel equ 125       * (30,45,144) auch mal
schrittweite equ 30  * probieren.

newpoly:                    * Programmstart
 move #schrittweite,d0      * Das Programm
 jsr @schreite              * wird endlos
 move #winkel,d0            * ausgefuehrt
 jsr @drehe                 * und ist nur
 move #schrittweite,d0      * durch RESET
 jsr @schreite              * zu stoppen
 move #winkel*2,d0
 jsr @drehe
 bra newpoly                * Sprung zurueck
end
```

Bild 23. Dieses Programm ...

Textstart=009000 Fenster=009000 Tor=009000 amer CTRL-J=Hilfe

Die Addition

Das Programm in *Bild 25* bringt wieder eine neue Anweisung. Es ist der Additionsbefehl, der hier verwendet wird, um ein Register um den Wert 2 zu erhöhen. Dazu wird hier eine Konstante vor der Schleife in das Register D3 geladen und dann in der Schleife bei jedem Durchlauf um 2 erhöht. Der Inhalt des Registers D3 wird als Schrittweite verwendet und dazu vor Aufruf des SCHREITE-Befehls mit MOVE D3,D0 in das Register D0 geladen.

Achtung, die Register D1 bis D2 sollte man nicht als Zwischenspeicher verwenden, da sie von den Unterprogrammen des Grundprogrammes zerstört werden könnten. D3 bis D7 sind dagegen frei verfügbar, solange man das Grundprogramm verwendet. Wenn man dabei aber den TRAP #1-Befehl verwendet, so kann man doch Register D7 nicht benutzen, da ja dort der Index geladen wird. *Bild 26* zeigt das Ergebnis des Pro-

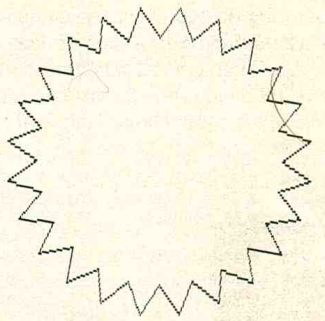


Bild 24. ... zeichnet einen Stern

```
seite equ 1          * Start-Seitenlaenge
winkel equ 121       * 95,181,120,121 ...
                     * versch. Figuren
polyspi:
 move #500-1,d4      * Gesamtzahl der
 move #seite,d3      * Wiederholungen
polyschleife:
 move d3,d0          * Seitenlaenge nach D0
 jsr @schreite       * dann damit Schreiten
 move #winkel,d0
 jsr @drehe
 add #2,d3           * jedesmal erhoehen
 dbra d4,polyschleife
 rts

end
```

Bild 25. Ein kleines Programm ...

Textstart=009000 Fenster=009000 Tor=009000 amer CTRL-J=Hilfe

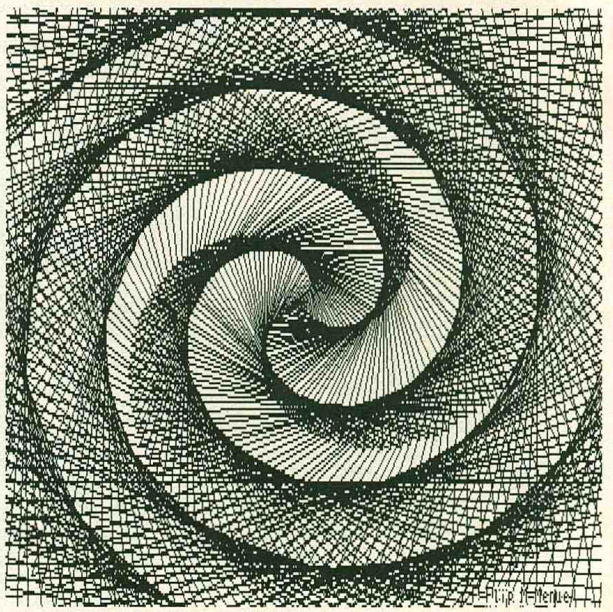


Bild 26. ... mit schöner Wirkung

```
fuell:
 move #256,d4    * X-Koordinate
 move #256,d5    * Y-Koordinate
 move #90,d6     * Winkel
 move #100,d7    * Breite
schleife:
 move d4,d1
 move d5,d2
 move d6,d3
 jsr @set        * Position
 move #100,d0
 jsr @schreite
 add #1,d4       * X jeweils erhoehen
 dbra d7,schleife
 rts

 end
```

Bild 27. Auch Flächen können gezeichnet werden

```
Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

```
radius equ 100  * in Punkte

kreise:
 move #(radius*1000)/358,d3
 move #200-1,d4          * 200 Mal
einkreis:
 move #36-1,d5
schleife:                * Einen  Kreis
 move d3,d0              * zeichnen
 jsr @schr16tel          * d3 ist Schritt-
 move #10,d0             * zahl
 jsr @drehe
 dbra d5,schleife        * innere Schleife
 sub #1000/358,d3        * eine Einheit
 dbra d4,einkreis        * kleiner
 rts

end
```

Bild 29. Das werden viele Kreise

```
Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

gramms am Bild 25. Man versuche auch hier unterschiedliche Werte bei der EQU-Anweisung einzusetzen und verändere auch einmal die Zahl beim Addiere-Befehl. Was geschieht?

Bild 28. Hier ein schwarzes Quadrat

F=Flip M=Menue

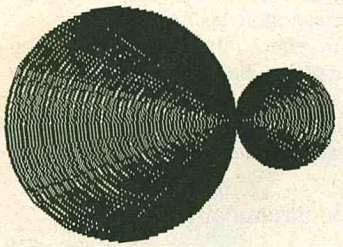


Bild 30. Hier sind diese vielen Kreise zu sehen

F=Flip M=Menue

Nun einmal ein Beispiel für den SET-Befehl aus dem Grundprogramm, den *Bild 27* zeigt. Damit kann man eine Fläche auf dem Bildschirm zeichnen. *Bild 28* zeigt das Ergebnis. In *Bild 29* noch ein praktisches Beispiel zur Erzeugung von Kreisen mit vorgegebenem Radius.

Das Programm erzeugt unterschiedlich große Kreise auf dem Bildschirm. Dazu wird die SCHR16TEL-Anweisung verwendet.

SUB

Ferner ein neuer Befehl des 68008, der SUB-Befehl, mit dem man einen Wert subtrahieren kann. Beim Subtraktions- und MOVE--Befehl wurde von der Möglichkeit, dem Assembler auch komplexe Ausdrücke angeben zu können, Gebrauch gemacht. Dabei ist wichtig, daß z. B. die Division nur bei der Übersetzung ausgerechnet wird und nicht später, wenn das Programm läuft. *Bild 30* zeigt das erzeugte Bild.

Aufgaben

1. Wie kann man die Formel aus Bild 29 herleiten? Dazu muß man die Formel „Umfang = 2 * PI * RADIUS“ und die Bedingung „16 Schritte mit SCHR16TEL ergeben einen Punkt“, sowie die Bedingung, daß man 36 Schleifendurchläufe für einen Kreis hat, ausnutzen.

2. Was passiert, wenn man zweimal den gleichen Namen in einem Programm mit unterschiedlichen Werten definiert?

3. Was passiert, wenn man einen unbekannten Namen im Programm verwendet?

4. Man probiere im Programm POLYSPI, anstelle der Seitenlänge den Winkel bei jedem Durchlauf zu erhöhen. Wie sehen die Figuren aus?

5. Was steht nach der Ausführung der folgenden Befehle im Register D0.L?

```
START:
    MOVE.L #$12345678,D0
    MOVE.W #$ABCD,D0
    MOVE.B #$EF,D0
    RTS
```

Dazu verwende man das Einzelschrittmenü.

6. Welche Aufgaben hat ein Assembler?

7. Was passiert, wenn man im Optionen-Menü die Debug-Information mit „3 = Debug Info an“ nicht einschaltet und dann im Einzelschritt ein Programm durchläuft?

Rolf-Dieter Klein

Schleifen und Roboter

Schrittmotore sind wichtige Elemente von industriellen Steuerungen und Maschinen. Sie werden zum Beispiel verwendet, um Roboter oder elektronische Zeichenmaschinen anzutreiben.

Ein Schrittmotor kann eine Anzahl fester Drehstellungen einnehmen. Dazu besitzt er einen Festmagneten auf der Achse des Schrittmotors und Elektromagnete, die zum Beispiel wie in *Bild 1* angeordnet sind. Mit den beiden Schaltern S1 und S2 wird bestimmt, welcher Strang der Wicklung der Elektromagneten unter Strom stehen soll.
In Bild 1a sind die Stränge P und R eingeschaltet. Durch die eingestellte Stromrichtung werden Nord- und Südpol am Stator erzeugt. Der Permanentmagnet auf der Achse des Schnittmotors richtet sich entsprechend aus. Wenn man jetzt S1 umschaltet, so ergibt sich die neue Orientierung der Magnetpole nach Bild 1b. Der Anker hat sich um 90 Grad gedreht, denn die Polarität am Stator PQ hat sich dessen Magnetfeld umgekehrt, weil der Strom durch die Spule Q in der Gegenrichtung gegenüber der Situation in Bild 1a läuft. Wenn man nun anschließend auch S2 umschaltet, so dreht sich der Motor wieder um 90 Grad entgegen dem Uhrzeigersinn. Will man ihn dann nochmals drehen, so muß man S1 wieder umschalten, dann S2 usw.

Hier stehen immer zwei Spulen unter Strom. Wenn man kleinere Drehwinkel erhalten will, so kann man die Stromkreise durch je einen weiteren Schalter auch noch wahlweise auftrennen, so daß dann nur eine Spule unter Strom steht. Mit der vorgeschlagenen Anordnung lassen sich dann 45-Grad-Stufungen erreichen.
Neben dem in Bild 1 gezeigten sogenannten Unipolar-Schrittmotor gibt es auch den bipolaren Schrittmotor. *Bild 2* zeigt dessen Schema. Zunächst fällt auf, daß die Mittelanzapfung der Spulen entfällt. Die Schalter sind nun so verdrahtet, daß man den Stromfluß durch die Spule umpolen kann. Damit läßt sich der gleiche Effekt wie beim unipolaren Motor erreichen. Allerdings stehen jetzt alle Spulen unter Strom. Damit ergibt sich ein besserer Wirkungsgrad. Bipolare Motore besitzen daher höhere Drehmomente als unipolare Motore. Wichtig! Unipolare Motore könnte man zwar auch so verschalten, wie es Bild 2 zeigt, doch erzeugen die Ströme in den Wicklungen auch Wärme, und unipolare Motore sind nicht dafür ausgelegt, alle Spulen ständig unter Strom zu halten. Da die Ansteuerschaltung für bipolare Motore komplizierter ist, werden wir hier einen Unipolar-Schrittmotor verwenden. *Bild 3* zeigt nochmals das Schema.

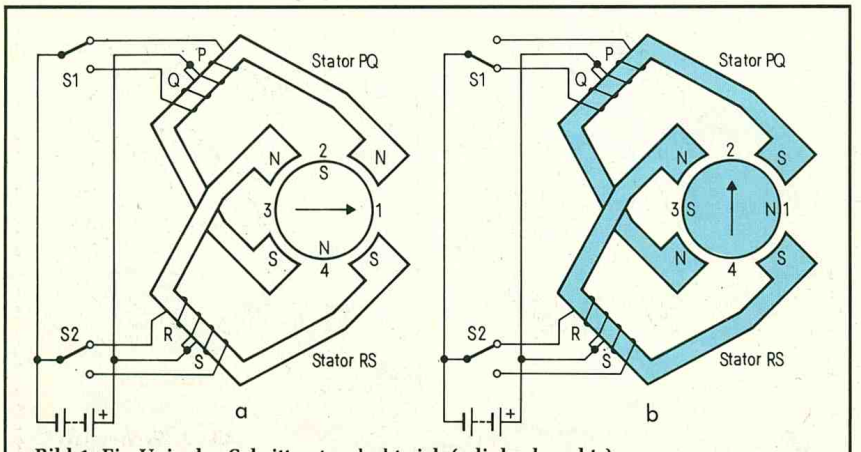


Bild 1. Ein Unipolar-Schrittmotor dreht sich (a links, b rechts)

Die Ansteuerung

Einen Schrittmotor kann man mit einfachen Leistungstransistoren ansteuern, *Bild 4* zeigt die komplette Schaltung (die übrigens auch auf der Roboter-Baugruppe enthalten ist). Man kann sich die Schaltung auf der IOE-Baugruppe selbst verdrahten. Zur Signaleingabe wird ein Tastenfeld mit 8 Tasten an den Eingang I/O_0 angelötet, das auch für spätere Versuche gebraucht wird. Mit diesen Tasten kann man den Schrittmotor per Programm steuern. Man könnte auch die normale Eingabetastatur verwenden, jedoch bringt das Tastenfeld den Vorteil, daß man mehrere Tasten gleichzeitig drücken kann, um so z. B. mehrere Funktionen gleichzeitig auszulösen.
Eine Aufgabe: Durch Drücken der Taste 0 soll sich der Motor in die Vorwärtsrichtung drehen, bei Drücken der Taste 1 rückwärts. Wie geht man hier bei der Programmentwicklung vor? Zunächst muß die Aufgabenstellung klar sein, was wohl der Fall ist. Dann folgt die Problemanalyse. Wie man den Motor bedienen muß, damit er sich dreht, ist klar; die Tasteneingabe ist ebenfalls klar. Jetzt kann man ein Struktogramm erstellen, das zunächst einmal grob den Programmablauf festlegt. Dann wird der Ablauf verfeinert und schließlich in ein Assemblerprogramm übertragen. Das Assemblerprogramm wird eingegeben, mit dem Assembler übersetzt und dann gestartet, wenn keine syntaktischen Fehler mehr vorliegen, die der Assembler bemängelt. Nun folgt der eigentliche Programmtest. Man sollte alle Kombinationsmöglichkeiten prüfen. Tritt ein Fehler auf, so sollte man zunächst das Struktogramm überprüfen, dann die Übertragung in das Assemblerprogramm. Man sollte seine Fehler systematisch suchen. So kann man einen nach dem anderen beseitigen. Doch nun zum Struktogramm, das uns noch fehlt. Dazu ein paar allgemeine Dinge vorab.

Struktogramme helfen

Man kann in jedem längeren Programm drei grundlegende Programmstrukturen finden. Einmal die einfache Verkettung von Befehlen (*Bild 5*), bei der ein Befehl hinter den nächsten gesetzt ist. Dann den Schleifenbaustein (*Bild 6* und *Bild 7*). Eine Anzahl von Befehlen wird so

lange wiederholt, bis eine Bedingung erfüllt wird, die zum Abbruch der Wiederholung führt. Dabei gibt es zwei unterschiedliche Schleifenformen. Bei der Form aus Bild 6 wird die Befehlssequenz in jedem Fall einmal durchlaufen, bevor die erste Abfrage erfolgt. Die Abfrage steht also am Schluß der Befehlssequenz. Oder wie in Bild 7. Dort wird zunächst die Bedingung solange abgefragt und nur, wenn diese Bedingung erfüllt ist, wird die Befehlssequenz durchlaufen. Bei der Form aus Bild 6 werden die Befehle innerhalb der Schleife immer mindestens einmal durchlaufen. Bei Bild 7, kann es sein, daß die Befehlssequenz überhaupt nicht durchlaufen wird, wenn die Anfangsbedingung nicht erfüllt ist.

Die dritte Strukturvariante ist der Verzweigungsbaustein. In Abhängigkeit von einer Bedingung wird im Ja-Fall eine andere Befehlssequenz durchlaufen, als im Nein-Fall. Damit lassen sich Entscheidungen realisieren.

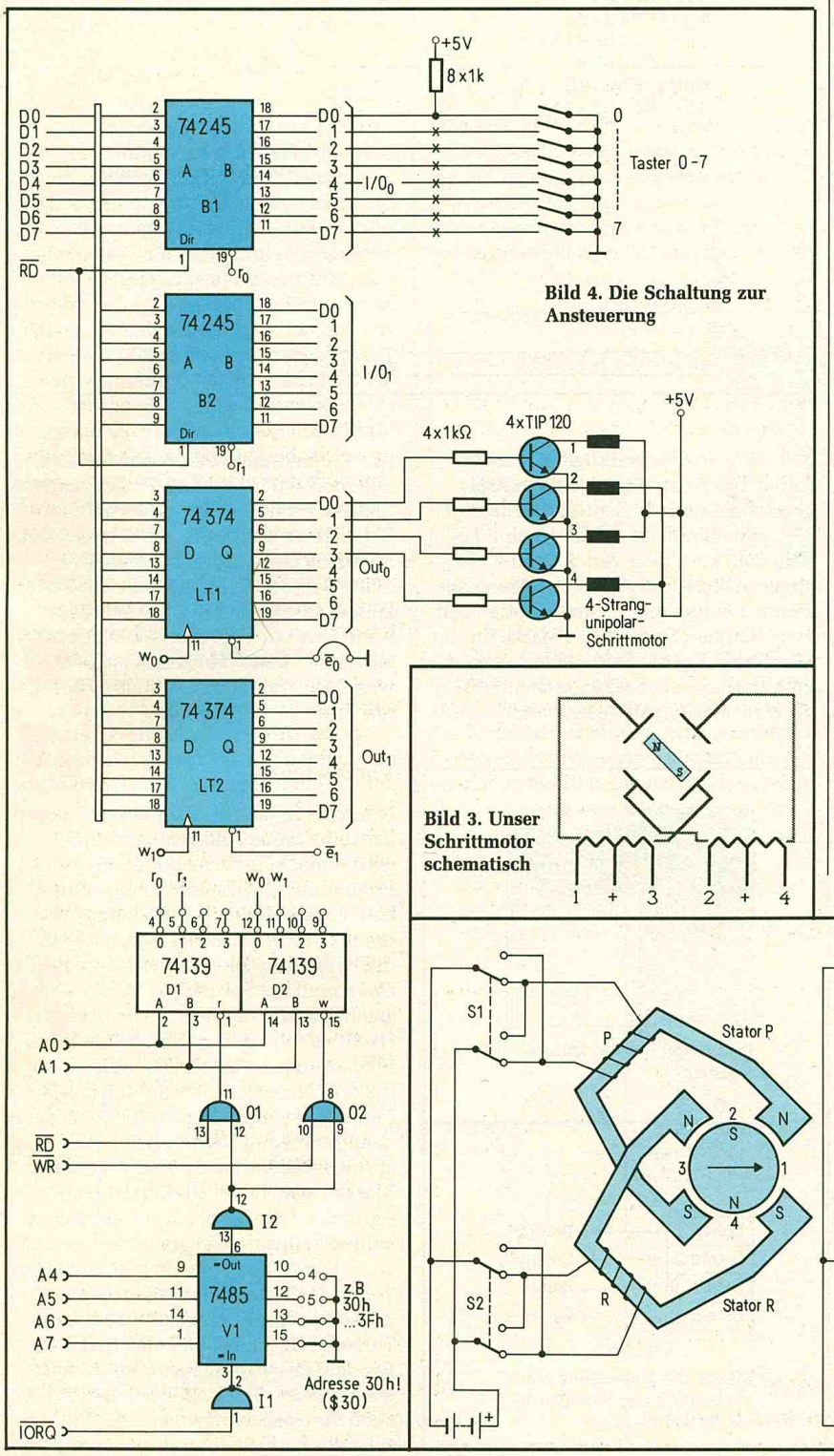


Bild 4. Die Schaltung zur Ansteuerung

Bild 3. Unser Schrittmotor schematisch

Die Grundstrukturen in 68008-Assembler

Bild 9 zeigt die Vorgehensweise bei einer Verkettung von Befehlen: Es wird einfach Befehl für Befehl in den Assembler übertragen.

Bild 10 zeigt ein Beispiel im Assemblercode. Es gibt dort keine Sprungbefehle, sondern nur Unterprogrammaufrufe und 68008-Anweisungen.

Beim Schleifenbaustein sieht die Realisierung anders aus. *Bild 11* zeigt das Umsetzungs-Schema. Zunächst beginnt man den Block mit der Definition einer Marke. Diese Marke kann einen beliebigen Namen haben, sofern er nicht mit einem anderen schon benutzten kollidiert. Dann wird der innere Block der Schleife programmiert. Dort können in „Unterblöcken" wieder alle drei Grundstrukturen vorkommen. Das wird deutlich, wenn man sich überlegt, daß anstelle eines Befehls auch ein korrekt programmiertes Struktogramm eingesetzt werden kann. Am Schluß erfolgt die Abfrage der Bedingung, deren Parameter natürlich bereitgestellt werden muß.

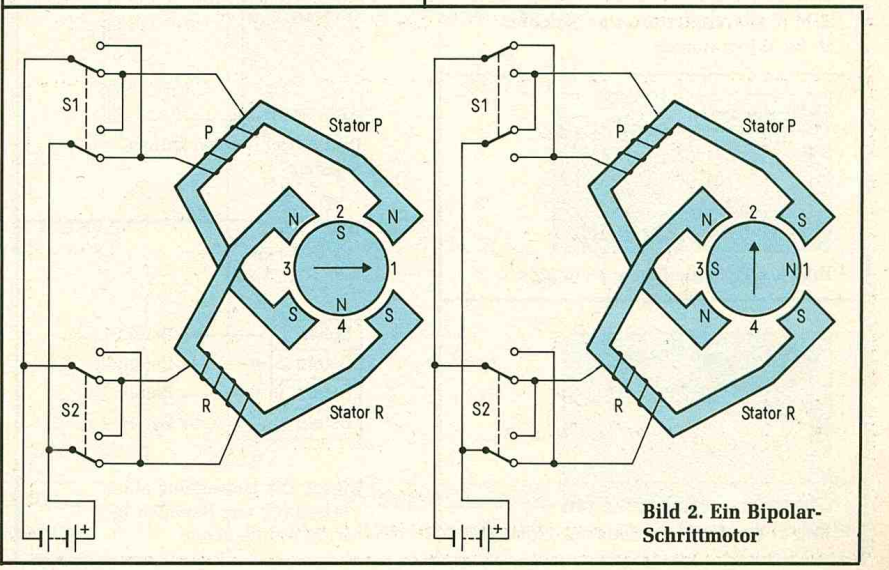


Bild 2. Ein Bipolar-Schrittmotor

Dann muß ein Sprung zurück zur Marke kommen, wenn die Bedingung nicht erfüllt ist. Wenn sie erfüllt ist, muß der Sprung übergangen werden und die Schleife verlassen werden, indem die nächste auf die Schleife folgende Anweisung absolviert wird.
Bild 12 zeigt ein einfaches Beispiel, das nach diesem Schema konstruiert ist. Es wird die Schildkröte dauernd entlang einer Kreisbahn geführt, bis eine Taste auf der Tastatur betätigt wird. Dazu werden zwei neue Befehle verwendet. Mit dem Unterprogramm CSTS, kann man die Bedingung „ist Taste gedrückt?" abfragen. Das Unterprogramm liefert den Wert FF im Register D0.B ab, wenn eine Taste gedrückt wurde, sonst den Wert 0. Gleichzeitig werden die sogenannten Bedingungsflags im Status-Register des 68008 gesetzt, die bei bedingten Sprüngen abgefragt werden. In diesem Fall durch BEQ (Branch if equal zero, springe, wenn das Zeroflag gesetzt). Dabei erfolgt der Sprung also, wenn das Null-Flag gesetzt, ist und das geschieht genau dann, wenn CSTS den Wert 0 im Register D0.B abgelegt hat. Wenn aber der Wert 0 vorliegt, so ist die Bedingung für den Schleifenabbruch nicht erfüllt, denn die Schleife soll ja erst beendet werden, wenn eine Taste gedrückt wird. Wenn diese Bedingung nicht erfüllt ist, so muß also die für den bedingten Sprung zur Marke geltende Bedingung erfüllt sein, die Schleife wird wiederholt.

```
▒
▒

* Verkettung von Befehlen
start:
 move #50,d0
 jsr @schreite
 jsr @hebe
 move #10,d0
 jsr @schreite
 jsr @senke
 move #50,d0
 jsr @schreite
 rts

 end

▒
▒
▒
▒
▒
▒
Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

Bild 10. Ein Assemblerbeispiel zur Verkettung

Nun können Sie das Programm leicht testen. Nach dem Start wird ein Kreis gezeichnet und die Schildkröte bewegt sich andauernd auf der Kreisbahn. Sobald man eine Taste drückt, bleibt sie stehen. *Bild 13* zeigt die Umsetzung der zweiten Schleifenart. Hier benötigt man zwei Marken. Einmal eine Marke für die Wiederholung der Schleife und eine, die zum Verlassen der Schleife dient. *Bild 14* zeigt ein Beispielprogramm.

Es ist nicht immer eindeutig, welcher Schleifentyp in welchen Programmiersituationen verwendet werden soll. Im Beispiel aus Bild 14 ist die Aufgabe fast identisch zum letzten Beispiel. Die Schildkröte soll entlang einer Kreisbahn laufen, solange keine Taste gedrückt wird. Hier wird die Schildkröte aber auch nicht den kleinsten Schritt tun, wenn das Programm bei einer gedrückten Taste gestartet wird.

Verzweigung

Nun zur letzten Art, dem Verzweigungsbaustein. Seine Umsetzung ist nicht ganz einfach, denn man muß die beiden Programmteile hintereinander schreiben. Für die gesamte Umsetzung benötigt man zwei Marken. Zunächst muß die Bedingung bereitgestellt werden. Dann muß ein Sprung zur Marke 1 folgen, wenn die Bedingung nicht erfüllt ist. Dann folgt der Ja-Teil, der bei NEIN übersprungen wird mit allen Befehlen, die dort nötig sind. Am Schluß des Ja-Teils steht ein Sprung zur Marke 2, denn man muß ja den Nein-Teil bei JA überspringen. Danach folgt jetzt hinter der Marke 1 der Nein-Teil und dann abschließend die Marke 2, bei welcher in beiden Fällen das Programm fortgesetzt wird. Es gibt auch die Möglichkeit einen Zweig des Verzweigungsbausteins leer zu lassen, dann stehen entweder im Ja-Teil oder im Nein-Teil keine Befehle. Bei der Übertragung ins Assemblerprogramm kann man sich in solchen Fällen auch Sprungbefehle einsparen. *Bild 16* zeigt ein Beispiel für einen ausprogram-

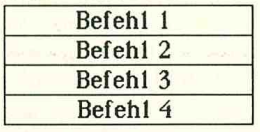


Bild 5. Die Verkettung von Befehlen in Struktogrammen

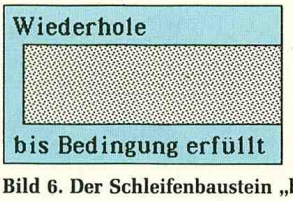


Bild 6. Der Schleifenbaustein „bis"

Bild 7. Der Schleifenbaustein „solange"

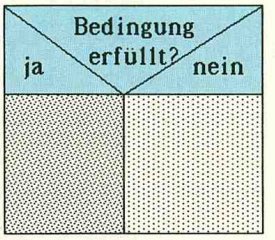


Bild 8. Der Verzweigungsbaustein

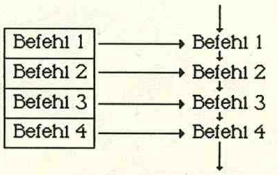


Bild 9. Die Umsetzung einer Verkettung von Befehlen in den Maschinencode

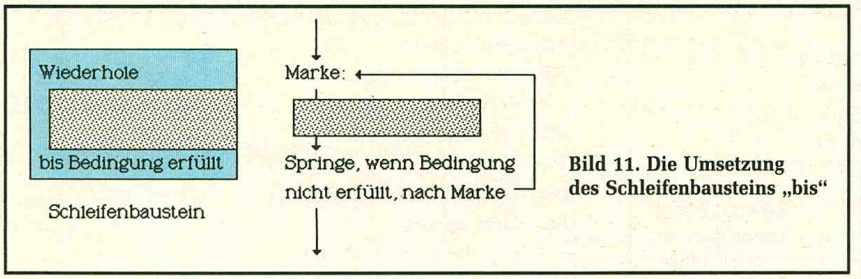


Bild 11. Die Umsetzung des Schleifenbausteins „bis“

mierten Verzweigungsbaustein. Ein Zeichen soll von der Tastatur gelesen werden. Wenn man das Zeichen A eingegeben hat, so soll eine senkrechte Linie auf dem Bildschirm erscheinen, sonst eine waagrechte Linie. Zum Einlesen eines Zeichens wird ein neues Unterprogramm benötigt. Die Routine CI (Console In) liest ein Zeichen von der Tastatur in das Register D0.B. Dabei wartet die Routine so lange, bis eine Taste gedrückt wird. Mit dem Befehl CMP.B wird der Vergleich durchgeführt. Dabei bewirkt der CMP-Befehl eine Subtraktion des links angegebenen Wertes vom rechten. Das Subtraktionsergebnis wird aber nirgendwo abgelegt, sondern es werden nur die Bedingungsflags gesetzt. Wenn also im Register D0.B der große Buchstabe A ankommt (also Shift-A drücken), so liefert die Subtraktion mit A den Wert 0 und der Sprung BNE (Branch non equal, also springe wenn nicht 0), wird nicht ausgeführt. Wenn man ein anderes Zeichen als A eingibt, so wird der Sprung ausgeführt, weil sich bei der Subtraktion ein Wert ungleich 0 ergibt.

Die Sprungbefehle

Bild 17 zeigt eine Zusammenstellung der bedingten Sprungbefehle, die Sie aber nicht alle gleich auf einmal verstehen müssen, sondern nacheinander kennenlernen sollen. Im 68008 gibt es mehrere Bedingungsflags. Einmal das Z-Flag, oder Zero-, Null-Flag genannt. Es ist gesetzt, wenn sich der Wert 0 als Ergebnis der letzten die Bedingungsflags setzenden Operation ergeben hat. Dann das N-, oder Negativ-Flag. Es wird gesetzt, wenn das Vorzeichenbit eines Ergebnisses gesetzt ist. Dann das C-, oder Carry-Flag. Es wird gesetzt, wenn ein Übertrag entstanden ist, z. B., wenn man die Zahl 5 von der Zahl 3 subtrahiert. Dann noch das V-, oder Overflow-Flag, das gesetzt wird, wenn ein arithmetischer Überlauf stattgefunden hat. Man darf es nicht mit dem C-Flag verwechseln. Diese vielen „Bedingungs-Flaggen“ werden von verschiedenen bedingten Sprüngen, auch in bestimmten Kombinationen, ausgewertet. Die ersten beiden Befehle der Tabelle sind uns schon bekannt. Sie prüfen das Z-Flag, also, ob der Wert 0 nach einer Operation vorgelegen hat oder nicht. Die nächsten vier Befehle dienen der Abfrage bei arithmetischen Operationen mit Zweierkomplementzahlen. So kann man zwei Zahlen miteinander vergleichen, wenn man vor dem Sprung den Befehl CMP verwendet. Die nächsten vier Befehle dienen dem Vergleich von Zahlen, die nicht im Zweierkomple-

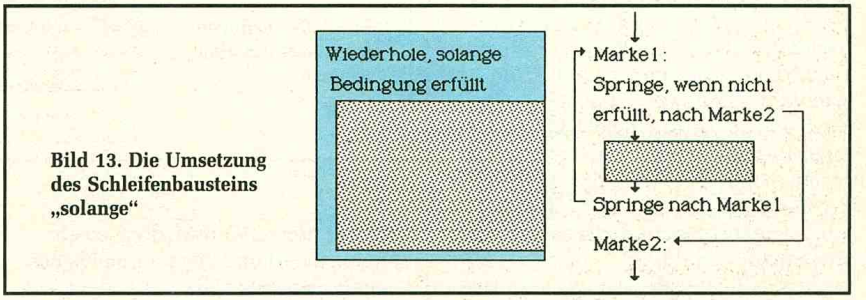


Bild 13. Die Umsetzung des Schleifenbausteins „solange“

```
*
*
*

* Der Schleifenbaustein

marke:          * Wiederhole
 move #1,d0     *    Befehl 1
 jsr @schreite  *    Befehl 2
 move #1,d0     *    Befehl 3
 jsr @drehe     *    Befehl 4
 jsr @csts      *  ( Bedingung erzeugen )
 beq marke      * bis Taste gedrueckt
 rts            * Ende

 end

*
*
*
*
*
Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

Bild 12. Ein Programmbeispiel zum Schleifenbaustein „bis“

```
*

* Der Schleifenbaustein

marke1:         * Wiederhole solange
 jsr @csts      * Taste nicht gedrueckt
 bne marke2
 move #10,d0    *    Befehl 1
 jsr @schreite  *    Befehl 2
 move #10,d0    *    Befehl 3
 jsr @drehe     *    Befehl 4
 bra marke1
marke2:
 rts            * Ende

 end

*
*
*
*
*
Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

Bild 14. Ein Programmbeispiel für den Schleifenbaustein „solange“

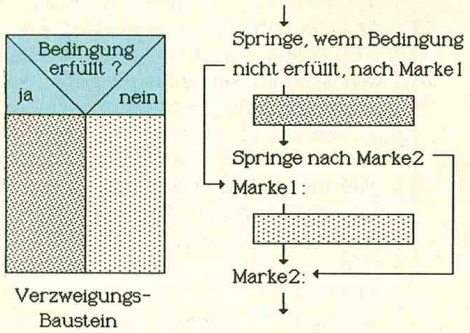


Bild 15. Die Umsetzung des Verzweigungsbausteins

BEQ	equal, gleich	Z
BNE	not equal, ungleich	$\overline{Z}$
BGT	greater than, größer	$N*V*\overline{Z}+\overline{N}*\overline{V}*\overline{Z}$
BGE	greater or equal, größer gleich	$N*V+\overline{N}*\overline{V}$
BLE	less or equal, kleiner gleich	$Z+N*\overline{V}+\overline{N}*V$
BLT	less, kleiner	$N*\overline{V}+\overline{N}*V$
BHI	high, mehr	$\overline{C}*\overline{Z}$
BHS	high or same, mehr oder gleich	$\overline{C}$
BLS	less or same, weniger oder gleich	C+Z
BLO	less, weniger	C
BVC	overflow clear, kein Überlauf	$\overline{V}$
BVS	overflow set, Überlauf	V
BPL	plus, positiv	$\overline{N}$
BMI	minus, negativ	N
BCC	carry clear, kein Carry	$\overline{C}$
BCS	carry set, Carry	C

Bild 17. Der bedingte Sprungbefehl in allen Varianten

ment dargestellt werden, also ohne Vorzeichen sind. Dann kann man auch einzelne Flags noch getrennt prüfen. Der Befehl BCC ist übrigens identisch zum Befehl BHS und der Befehl BCS identisch mit BLO; die beiden unterschiedlichen Namen wurden eingeführt, um Assemblerprogramme übersichtlicher zu gestalten. So weiß man bei BLO, daß man einen arithmetischen Vergleich abfragen will und bei BCS, daß man nur das Carry-Flag prüfen will (z. B. nach einer Schiebeoperation, die nichts mit Arithmetik zu tun hat).
Nicht alle Befehle des 68008 verändern die Flags. Das muß so sein, denn oft ist eine Abfrage erst dann sinnvoll, wenn zwischen Herstellung einer Bedingung und Abfrage noch einige Befehle liegen (zum Abspeichern im Memory zum Beispiel). Es verändern aber alle MOVE-Befehle (außer MOVEM, doch das kommt später) und alle arithmetischen Operationen, wie SUB, ADD oder CMP, die Flags.
Bild 18 zeigt noch ein Beispiel für eine Auswertung mehrerer Flags. Bis auf die Befehle SETPEN und ERAPEN sind alle Befehle bekannt. ERAPEN schaltet den Grafik-Prozessor auf den Löschmodus, alle Linien, die dann gezeichnet werden, werden schwarz geschrieben und nach SETPEN wieder weiß.
Hinweis: Bei älteren Grundprogrammen (3.2) kann es vorkommen, daß bei Ausgabe der Schildkröte wieder auf SETPEN geschaltet wird, also alle 20 ms. Wenn man das verhindern will, so muß man mit HIDE die Schildkröte zunächst aus-

```
▒
▒
▒

* Der Verzweigungsbaustein

start:              * Bitte Zeichen eingeben
 jsr @ci            * Zeichen als Bedingung
 cmp.b #'A',D0      * Wenn Der Buchstabe A
 bne marke1         * Dann JA-Teil
 move #100,d0       * und Linie zeichnen
 jsr @schreite      * die Senkrecht ist
 bra marke2         * Nein-Teil überspringen
marke1:
 move #90,d0        * Sonst waagrechte Linie
 jsr @drehe         * zeichnen
 move #100,d0
 jsr @schreite      * im Nein-Fall
marke2:
 rts                * Ende

 end

▒
Textstart=009000 Fenster=009000 Tor=009000      amer CTRL-J=Hilfe
```

Bild 16. Ein Programmbeispiel zum Verzweigungsbaustein

```
start:              * Wiederhole
 move #100,d0       *  Linie zeichnen
 jsr @schreite
 move #-100,d0
 jsr @schreite      *
 jsr @csts          *  Bedingung einlesen
 beq keinetaste     *  Wenn Taste gedrueckt
 jsr @erapen        *  erst mal alte Position
 move #100,d0       *  löschen
 jsr @schreite      *
 move #-100,d0
 jsr @schreite      *  dann Taste abfragen
 jsr @ci            *  dazu einlesen
 cmp.b #'R',d0
 bne warnichtr      *  Rechts
  move #-10,d0      *  also z.B. -10 laden
  bra wardochr
warnichtr:
 move #10,d0        *  also Links
wardochr:
 jsr @drehe         *  ausfuehren
 jsr @setpen        *  wieder auf Schreiben
keinetaste:         *  sonst nichts weiter
 bra start          * endlos

Textstart=009000 Fenster=009000 Tor=009002      amer CTRL-J=Hilfe
```

Bild 18. Ein kombiniertes Beispiel

blenden und sie mit SHOW nach dem Löschteil wieder einblenden.

Das Struktogramm wird verfeinert

Nun zum Schrittmotor. *Bild 19* zeigt das Struktogramm. Dabei beginnt das Struktogramm mit der Anweisung: Voreinstellungen durchführen. Dort müssen z. B. Variable initialisiert werden. Dann soll das Programm immer laufen, also folgt ein Schleifenbaustein, der aber unbegrenzt laufen soll, also dessen Solange-Bedingung immer erfüllt ist. In der Schleife wird dann zunächst die Taste 0 abgefragt. Wenn sie gedrückt ist, so wird in den Ja-Teil gesprungen. Wenn nicht, in den Nein-Teil. Dort wird eine weitere Unterscheidung vorgenommen. Wenn im Ja-Teil der Taste 1, die Taste 1 auch gedrückt wird, so liegt ein Fehler vor, und es soll keine Aktion stattfinden. Wenn in diesem Fall Taste 1 nicht gedrückt ist, so soll der Motor vorwärts laufen. Wenn Taste 0 nicht gedrückt ist, wohl aber Taste 1, so soll der Motor rückwärts laufen. Wenn keine Taste gedrückt ist, so soll nichts passieren. Man sieht, daß man im Struktogramm besonders schön alle Fehlerfälle erfassen kann.

Nun kann man für die Teilprobleme wieder Struktogramme erstellen. Und das Ganze verfeinern. *Bild 20* zeigt das Assemblerprogramm, das am Schluß herauskommen sollte, wenn Sie sich an die Empfehlungen halten, die oben zur Programmentwicklung gegeben wurden. Es sei gesagt, daß die Aufgabe nicht einfach zu lösen ist, insbesondere, wenn man noch sehr wenige 68008-Befehle kennt. Also nicht verzagen, sondern Befehle lernen. Dazu ist z. B. das Buch „M68000-Familie, Teil 1" von Werner Hilf und Anton Nausch (tewi-Verlag) hilfreich, das leider 79 DM kostet.

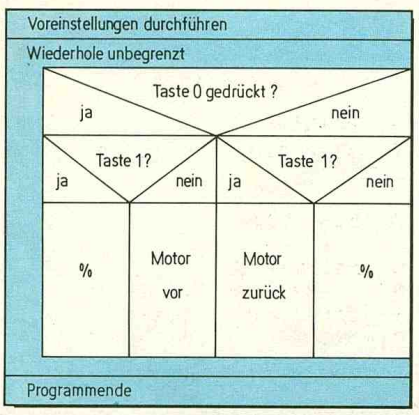


Bild 19. Das Struktogramm zum Schrittmotor

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1


009C00                  ;***********************************
009C00                  ; SCHRITTMOTORPROGRAMM VERSION 1.0  ;
009C00                  ; VOM 13.12.1983                    ;
009C00                  ;***********************************
009C00
009C00                  ; ******** VEREINBARUNGS-BAUSTEIN ********
009C00                  ;
= 00000001              MAG1    EQU 1   ; MOTORBITS, EINZELNE MAGNETE
= 00000002              MAG2    EQU 2   ; BITZUORDUNGSTABELLE
= 00000004              MAG3    EQU 4
= 00000008              MAG4    EQU 8
009C00
= 00000001              VOR     EQU 1   ; TASTENBIT-ZUORDNUNG
= 00000002              RUECK   EQU 2
009C00
= FFFFFF30              MOTOR   EQU $FFFFFF30
= FFFFFF30              TASTEN  EQU $FFFFFF30
009C00
009C00                  TAB:                    ; SCHALTSEQUENZ-TABELLE
009C00  01                      DC.B MAG1
009C01  03                      DC.B MAG1+MAG2
009C02  02                      DC.B MAG2
009C03  06                      DC.B MAG2+MAG3
009C04  04                      DC.B MAG3
009C05  0C                      DC.B MAG3+MAG4
009C06  08                      DC.B MAG4
009C07  09                      DC.B MAG4+MAG1
009C08
009C08                  ;  ******** UNTERPROGRAMM-BAUSTEIN *********
009C08
009C08                  MOTORVOR:
009C08  5201                    ADDQ.B #1,D1    ; EIN SCHRITT VORWAAERTS
009C0A                                          ; GEMEINSAMER TEIL VOR,RUECK
009C0A  0201 0007       MOTORW: AND.B #7,D1     ; BEREICH 0..7 IST ERLAUBT
009C0E  13F0 1000               MOVE.B 0(A0,D1.W),MOTOR ; AUSGABE AN DEN MOTOR
009C12  FFFFFF30
009C16                                          ; SCHRITTGESCHW. DURCH
009C16  203C 000007D0           MOVE.L #2000,D0 ; WARTESCHLEIFE
009C1C  4E71            WARTE:  NOP             ; FUER ZEITVERZOEGERUNG
009C1E  51C8 FFFC               DBRA D0,WARTE
009C22  4E75                    RTS
009C24
009C24                  MOTORRUECK:
009C24  5301                    SUBQ.B #1,D1    ; EIN SCHRITT RUECKWAERTS
009C26  60E2                    BRA.S MOTORW    ; WEITER WIE OBEN
009C28
009C28                  ; ******** PROGRAMM-BAUSTEIN ************
009C28
009C28  207C 00009C00  START:  MOVEA.L #TAB,A0  ; TABELLENSTART DEFINIEREN
009C2E  323C 0000               MOVE #0,D1      ; MOTOR POSITION ANNEHMEN
009C32  4239 FFFFFF30           CLR.B MOTOR     ; MOTOR AUS
009C38                                          ; WIEDERHOLE UNBEGRENZT
009C38  1039 FFFFFF30   WDH1:   MOVE.B TASTEN,D0 ; EINLESEN DES PORTS
009C3E  0200 0001               AND.B #VOR,D0   ; VOR-TASTE
009C42  6716                    BEQ.S TASTEVJ   ; KONTAKT SCHLIESST BEI 0
009C44
009C44  1039 FFFFFF30           MOVE.B TASTEN,D0 ; NOCHMALS EINLESEN
009C4A  0200 0002               AND.B #RUECK,D0
009C4E  6622                    BNE.S ENDE      ; NICHT GEDRUECKT
009C50
009C50  4EB9 00009C24           JSR MOTORRUECK  ; MOTOR RUECK
009C56  6000 001A               BRA ENDE
009C5A
009C5A                  TASTEVJ:                ; WENN VORW. DANN
009C5A  1039 FFFFFF30           MOVE.B TASTEN,D0
009C60  0200 0002               AND.B #RUECK,D0 ; NICHT ZUGLEICH RUECKW.
009C64  6700 000C               BEQ ENDE        ; FEHLERFALL
009C68
009C68  4EB9 00009C08           JSR MOTORVOR    ; MOTOR EINEN SCHRITT VOR
009C6E  6000 0002               BRA ENDE
009C72
009C72  6000 FFC4       ENDE:   BRA  WDH1       ; UNBEGRENZT DURCHFUEHREN
009C76
009C76                  ; ***************************************
009C76                          END

008B80  Ende-Symboltabelle
```

Bild 20. Das Assemblerlisting des Schrittmotorprogramms

Das Schrittmotor-Steuerprogramm

Zunächst folgt der Vereinbarungsbaustein, so hat es sich als zweckmäßig erwiesen. Die einzelnen Bits, die man zum Einschalten der Spulen benötigt, werden mit EQU-Anweisungen definiert, so daß man sie bequem beim Namen nennen kann. Magnet 1 soll an Bit 0 angeschaltet werden, dessen Stellenwert 1 ist, Magnet 2 kommt an Bit 1, dessen Stellenwert 2 ist usw. Damit kann man später die Magnete einfach ein- und ausschalten, indem man ins I/O-Register 1, 2, 4 oder 8 schreibt, was durch die Nennung des entsprechenden Namens MAG 1 geschieht. Wird am Port $FFFFFF30 der Wert 1 ausgegeben, so wird Magnet 1 eingeschaltet, wird der Wert 2 ausgegeben, wird Magnet 2 eingeschaltet, mit 4 der Magnet 3 und mit 8 der Magnet 4. Nun kann man mit dem Menü IO-Setzen den Motor bewegen, indem man selbst von Hand die richtigen Kombinationen ausgibt. Dabei kann man den Motor bewegen, indem man immer zwei Spulen ansteuert, oder indem man mit dem sogenannten Halbschrittverfahren, auch einzelne Spulen verwendet. Im Bild 20 ist die Tabelle für die Bewegung bei der Marke TAB abgelegt. Wenn man die Werte, die der Assembler auf der linken Hälfte des Listings ausgedruckt hat, eingibt, so dreht sich der Motor. Also gibt man auf den Port $FFFFFF30 nacheinander die Werte 1, 3, 2, 6, 4, $C, 8, 9. Dreht sich der Motor dabei nicht ruckfrei, so haben Sie einen Verdrahtungsfehler eingebaut, vielleicht sind Wicklungen vertauscht oder etwas ähnliches.

Adreßregister werden benutzt

Im Unterprogramm-Baustein wird die automatische Fortschaltung durchgeführt. Dazu wird nun erstmals ein Adreßregister verwendet. Das Adreßregister A0, wird im Programm-Baustein dazu mit der Adresse der Tabelle geladen. Dies geschieht mit dem Befehl MOVEA.L #TAB,A0. Im Unterprogramm-Baustein MOTORVOR und MOTORRUECK wird darauf zurückgegriffen. Nun benötigt man noch einen Zähler, der dafür sorgt, daß nacheinander die einzelnen Einträge der Tabelle durchlaufen werden. Dies geschieht im Register D1. Dazu wird beim Vorlauf der Wert 1 addiert und beim Rückwärtslauf der Wert 1 subtrahiert. Die Tabelle ist aber nur 8 Einträge groß und so wird mit dem Befehl AND.B #7,D0 erreicht, daß der Zähler nach Erreichen des Wertes 7 auf 0 springt, oder daß bei der Subtraktion nach 0 die Zahl 7 kommt. Der Befehl AND bewirkt eine UND-Verknüpfung. Man kann damit einfache, z. B. Binärzähler realisieren.

Nun kommt ein toller Befehl: MOVE.B 0(A0,D1.W),MOTOR. Es handelt sich um eine neue Adressierart beim MOVE-Befehl, die bedeutet: Nimm den Inhalt des Adreßregisters und addiere den Inhalt des Registers D1.W (also nur 16 Bit) hinzu. Dann addiere den Wert 0 (die Zahl vor der Klammer). Dieser Wert ist eine Adresse. Von dort hole ein Byte (wegen .B bei MOVE ist es ein BYTE) und schaffe sie zur Adresse MOTOR (MOTOR ist die Portadresse auf der IOE-Baugruppe). Damit wird also der neue Code an den Port ausgegeben. Nun folgt noch ein Teil der mit WARTE: anfängt. Es handelt sich um eine Verzögerungsschleife. Denn eines hatten wir bei der Problemanalyse vergessen: die Geschwindigkeit des Motors wurde nicht festgelegt. Der Schrittmotor dreht sich so schnell, wie die einzelnen Spulen umgeschaltet werden, aber nicht beliebig schnell. Daher muß die Umschaltgeschwindigkeit begrenzt werden und dies geschieht in der Warteschleife. Der Befehl NOP und DBRA wird 2001mal durchlaufen und dadurch ergibt sich eine Verzögerung des Programmlaufes, der ausreicht, die Motorgeschwindigkeit zu begrenzen. Durch Ändern des Wertes kann man die Drehzahl verändern.

Das Hauptprogramm ist nicht so aufregend, es entspricht dem Struktogramm. Eine Besonderheit ist noch die Abfrage der Tasten. Durch den Befehl MOVE.B TASTEN,D0 wird der Port, an dem die Tasten angeschlossen sind, als Byte ins Register D0.B übertragen. Will man eine einzelne Taste abfragen, so muß man zunächst die Bitstelle aus dem Byte isolieren. Dies kann man mit dem AND-Befehl tun (oder mit dem Befehl BTST, siehe 68008-Befehle). Durch AND.B #VOR,D0 wird eine Und-Verknüpfung mit dem Wert 1 (VOR EQU 1) durchgeführt. Wenn die Taste gedrückt wird, so gelangt ein 0-V-Pegel an den Port und das entsprechende Bit ist 0. Durch die Und-Verknüpfung ist das Register 0, wenn die Taste gedrückt wird (also BEQ springt) und wenn die Taste nicht gedrückt wird ist das Register D0.B mit einem Wert ungleich 0 geladen (also BNE springt, aber BEQ nicht). So kann man jede Bitstelle prüfen.

Die Adressierungsarten des 68000

Bild 21 zeigt noch eine Zusammenstellung der möglichen Adressierarten beim 68008. Er besitzt eine Vielzahl von Möglichkeiten, die man aber keinesfalls sofort alle nutzen muß. Das kommt nach und nach, wenn eigene Programme zunehmend eleganter werden.

Aufgaben

1. Man zeichne ein Struktogramm für die Unterprogramme des Unterprogramm-Bausteins.
2. Warum wird die Warteschleife 2001mal durchlaufen und nicht 2000mal, obwohl der Wert 2000 in das Register D0 geladen wird?
3. Was bewirkt das Attribut „.S" beim BRA, BEQ, BNE...-Befehl? Was ändert sich, wenn man es wegläßt? Achtung! Dazu die Assemblerausgaben vergleichen, bei der Programmausführung ergibt sich keine Änderung!
4. Was geschieht, wenn man bei der ADD- und SUB-Anweisung den Wert 2 addiert? Ausprobieren!

Syntax	Beispiel	Name
Dn	D0	Datenregister
An	A3	Adressregister
(An)	(A2)	Indirekt
(An)+	(A7)+	Postincrement
-(An)	-(A7)	Predecrement
d(An)	56(A0)	Displacement
d(An,Xn)	45(A0,D4.L)	Index mit Displ.
Abs.W	$1234.W	Absolut, kurz
Abs.L	$88991122.L	Absolut, lang
d(PC)	START(PC)	PC-Relativ
d(PC,Xn)	GEHT(PC,A3.W)	PC-Relativ, Index
Imm	#5422	Direkt

Bild 21. Die Adressierarten beim 68008

Theorie und Praxis rund um den NDR Klein-Computer

Vorkenntnisse

Wenn Sie Ihren NDR-Klein-Computer wirklich selbst bauen (nicht nur zusammenbauen) wollen, dann müssen Sie zweierlei beherrschen:

Zum einen das Löten. Saubere Lötstellen sind das A und O funktionierender Schaltungen. Insbesondere elektronische Bauelemente stellen an Lötfertigkeit und Lötkenntnisse höchste Ansprüche. Der Lehrgang „Lötpraktikum" vermittelt Ihnen die nötigen Fertigkeiten und Kenntnisse, um diesen Anforderungen gerecht zu werden.

Zum anderen benötigen Sie Elektronik-Grundkenntnisse, die Sie sich mit dem Kompakt-Kurs Elektronik bequem in 3 Monaten aneignen können.

Lötpraktikum

Lehrmaterial und Bauteilesortiment für 10 Lötübungen sowie Temperaturwächterschaltung.

DM 88,— · Bestell-Nr. 016

Mit Werkzeug (Lötkolben, Seitenschneider, Schraubendreher, Flachzange und Pinzette): DM 112,— · Bestell-Nr. 017

Kompakt-Kurs Elektronik

Lehrmaterial (186 Seiten A4) mit Experimentiermaterial, Tonbandkassette und Flipchart. DM 248,— · Bestell-Nr. 024

Beide Kurse wurden vom Technischen Lehrinstitut Christiani in Konstanz entwickelt und sind bereits von ca. 1900 Teilnehmern erfolgreich durchgearbeitet worden.

Begleitmaterial zur Sendung

Mikrocomputerpraxis mit dem NDR-Klein-Computer ist der Titel eines Lehrgangs, der exakt auf die Fernsehsendung zugeschnitten ist und im Sommer '85 vorliegen wird.

Hier eine vorläufige Inhaltsübersicht: Hardware-Beschreibung · Programmentwicklungen · Beispielprogramme · Erläuterung des Befehlssatzes · Aufgaben, Übungen.

In der Fernsehsendung wird die Computersprache BASIC nicht behandelt. Für diejenigen, die sich trotzdem damit beschäftigen wollen, gibt es den Kompakt-Kurs BASIC, der an den NDR-Klein-Computer angepaßt wurde.

Kompakt-Kurs BASIC

Lehrmaterial (ca. 200 Seiten A4) mit Tonbandkassette · DM 198,— · Bestell-Nr. 061

Sie lernen in kürzester Zeit Ihren Home-Computer in BASIC zu programmieren. Alle gängigen BASIC-Anweisungen werden erläutert und ihre Anwendung in vielen Beispielprogrammen demonstriert. Sie lernen spielend leicht, Probleme in Programme umzusetzen.

Christiani

Hier abtrennen und im Umschlag einsenden an: Dr.-Ing. P. Christiani GmbH, Techn. Lehrinstitut und Verlag, Postfach 35 69189, 7750 Konstanz

Bestellcoupon

- ☐ Lötpraktikum ohne Werkzeug DM 88,—
- ☐ Lötpraktikum mit Werkzeug DM 112,—
- ☐ Kompakt-Kurs Elektronik DM 248,—
- ☐ Kompakt-Kurs BASIC (f. d. Klein-Computer) DM 198,—
- ☐ Kostenloses Informationsmaterial zum Lehrgang Mikrocomputerpraxis mit dem NDR-Klein-Computer
- ☐ Kostenloses Informationsmaterial zu weiteren Veröffentlichungen aus dem Hause Christiani

Name, Vorname

Straße

PLZ, Ort

Datum Unterschrift

69189

Rolf-Dieter Klein

Die Landung auf dem Mond

Videospiele haben in den Wohnzimmern Einzug gehalten. Es fing alles ganz harmlos an. Zuallererst gab es eine Art Tennisspiel. Man mußte mit einem Schläger einen Ball in das gegnerische Feld hineintreiben. Damals waren die Spiele noch einfach gehalten. Dann explodierte die Entwicklung, es kamen die Simulationsspiele wie Mondlandung und auch Schießspiele und zuletzt die Abenteuerspiele.

In dieser Folge soll einmal gezeigt werden, wie ein einfaches Simulationsspiel aufgebaut ist. Dazu wurde die Mondlandung gewählt, da sie auch einige interessante physikalische Aspekte enthält und weil man sie mit der Grafik des NDR-Klein-Computers sehr hübsch gestalten und später auch ausbauen kann. Die Aufgabe: Eine Mondlandefähre soll sicher auf dem Mondboden aufgesetzt werden. Die Steuerung der Mondlandefähre soll durch Schubveränderung in vertikaler Richtung erfolgen können. Ferner soll die Mondlandefähre auch Steuerdüsen, die in horizontaler Richtung wirken, benutzen.
Professionals werden nun nicht gleich „losprogrammieren", sondern ein Pflichtenheft des zu erarbeitenden Programmes aufstellen. So ein Pflichtenheft ist nämlich eine wichtige Grundlage für die Programmierung. Es enthält alle Wünsche und Pflichten, die das auszuarbeitende Programm später erfüllen soll. Im Pflichtenheft wird also die Programmieraufgabe exakt definiert.

Pflichtenheft

1. Mondoberfläche darstellen
2. Ebener Landeplatz
3. Landefähre
4. Fähre beweglich
5. Steuerdüsen
6. Anziehungskraft berücksichtigen
7. Ergebnis anzeigen

5. Steuerdüsen

5.1 Düse nach links
5.2 Düse nach rechts
5.3 Düse nach oben
5.4 Düse nach unten
5.5 Vier Tasten zur Steuerung
5.6 Treibstoff begrenzen

Bild 1. Das Pflichtenheft zur Mondlandung

Das Pflichtenheft

Unser Pflichtenheft (*Bild 1*) gestaltet sich recht schlicht. Es besteht aus zwei Teilen. In der oberen Hälfte sind die Hauptpunkte aufgeführt und in der unteren ist eine weitere Detaillierung angegeben.

Zunächst soll eine Mondoberfläche dargestellt werden. Dies kann man später z. B. mit der Logo-Grafik tun. Dann soll ein ebener Landeplatz zur Verfügung gestellt werden, denn die Fähre muß ja auf der Mondoberfläche sauber aufsetzen können, sie darf nicht am Rande eines Kraters landen müssen. Dann benötigen wir natürlich die Landefähre selbst, die beweglich sein muß. Man könnte sie ebenfalls mit der Schildkrötengrafik konstruieren, jedoch werden wir eine schnellere und einfachere Methode zur Herstellung bewegter Grafiken kennenlernen. Um die Fähre zu bewegen, gibt es vier Steuerdüsen: Düse links, Düse rechts, Düse oben und Düse unten. Bei allem muß die Anziehungskraft des Mondes berücksichtigt werden. Das Ergebnis eines Landeversuches muß am Ende einer Spielaktion auf dem Bildschirm angezeigt werden. Vier Tasten sollen der Steuerung dienen, der vorhandene Treibstoff soll begrenzt sein, denn die Mondlandefähre ist klein.

Bewegung ins Spiel

Bevor Sie jetzt ein Struktrogramm entwerfen, müssen ein paar grundsätzliche Überlegungen angestellt werden. Zunächst die Formeln für die Bewegung. Zum einen kann sich die Fähre in X-Richtung bewegen und zum anderen in Y-Richtung. Da diese beiden Bewegungen unabhängig voneinander ablaufen können müssen, ergeben sich zwei Formeln, die allerdings viel miteinander zu tun haben. Jeder von Ihnen kennt die Formel für den Weg x, der bei einer bestimmten Geschwindigkeit v in einer bestimmten Zeit t zurückgelegt wird: $x = v \cdot t$. Setzt man einen Zeitpunkt einmal als Anfang der Betrachtung ein und nennt ihn t_o, dann kann man für zwei Zeitpunkte t_a und t_b, die von t_o an gemessen sind, die Wege x_a und x_b berechnen. In der Zeit zwischen t_a und t_b bewegt sich der betrachtete Körper von x_a nach x_b und legt also die Strecke $x_b - x_a$ zurück. Wenn der Körper dies mit einer bestimmten Geschwindigkeit tut, sie sei einmal mit $v_{a,b}$ bezeichnet, dann gilt

$$x_b - x_a = v_{a,b} \cdot (t_b - t_a).$$

Da Computer am bequemsten mit diskreten Schritten rechnen können, ist die obengenannte Formel schon ganz gut zur Beschreibung der Mondfährenbewegung in x-Richtung zu gebrauchen. Solange nämlich die Geschwindigkeit konstant bleibt, kann man für eine Reihe von Zeitpunkten, beginnend bei t_o und endend bei t_n folgendes aufschreiben:

$$x_n - x_o = v_{o,n} \cdot (t_n - t_o) =$$
$$v_{o,1} \cdot (t_1 - t_o) + v_{1,2} \cdot (t_2 - t_1) +$$
$$+ v_{2,3} \cdot (t_3 - t_2) + \ldots + v_{n-1,n} (t_n - t_{n-1})$$

Wählt man die aufeinanderfolgenden Zeitpunkte bei dieser Zerlegung des Zeitintervalles von t_o bis t_n alle mit gleich kleinem Abstand Delta $t = t_1 - t_0$, dann läßt sich die Änderung des Weges nach Ablauf von Delta t mit Delta $x = v_x$. Delta t ausrechnen. Bei konstantem v ist $v_x = v_o$ und zur Berechnung der x-Koordinate der Raumfähre muß immer nur die Konstante Delta $x = v_o \cdot$ Delta t nach jedem Delta t dem bisherigen x-Wert hinzugezählt werden. Jetzt müssen Sie sich aber festhalten, wenn zu irgendeinem Zeitpunkt die Düsen eingeschaltet werden. Denn dann wächst die Geschwindigkeit während der Zeit Delta t um Delta v an. Bei konstanter Beschleunigung b um den Betrag b · Delta t. Die Rechnung oben kann dann so geschrieben werden:

$$x_n - x_o = v_{o,1} \cdot \text{Delta t} +$$
$$+ (v_{o,1} + b \cdot \text{Delta t}) \cdot \text{Delta t} +$$
$$+ (v_{o,1} + b \cdot \text{Delta t} + b \cdot \text{Delta t}) \cdot \text{Delta t} + \ldots$$

Das gilt aber nur, wenn die Werte Delta t so klein genommen sind, daß die Geschwindigkeitsänderung innerhalb von Delta t praktisch nicht fühlbar ist. Bezeichnet man ($v_{0,1}$ + b · Delta t) einmal mit $v_{1,2}$, dann sieht man, daß das obengenannte Delta x in diesem Zeitabschnitt vom Delta x im Zeitabschnitt zwischen 0 und 1 um die Konstante b · (Delta t) verschieden ist. Je zwei aufeinanderfolgende Delta x sind also um eine von der Beschleunigung abhängige Konstante verschieden, das kann man an der Formel oben ablesen. Während oben die unbeschleunigte Bewegung in x-Richtung durch die fortlaufende Addition eines konstanten Delta x berechnet werden konnte, muß bei der beschleunigten Bewegung bei der Berechnung der neuen Position aus der alten das Delta x noch vor seiner Addition um eine Konstante geändert werden.

Schwerkraft

Das gilt nun alles analog auch für die y-Richtung. Dort kommt noch die Schwerkraft des Mondes hinzu, die sich einfach dadurch zeigt, daß Delta y während eines jeden Delta t um einen negativen Betrag wächst. Da die Natur wunderbarerweise so geartet ist, daß sich zwei Beschleunigungen so überlagern, daß man sie addieren kann um die Gesamtbeschleunigung daraus zu errechnen, kann man der Fallbeschleunigung auf dem Mond durch Addition einer Beschleunigung durch Rückstoßdüsen entgegenwirken. Dem Delta y werden also zwei Konstanten, die Fallbeschleunigung und die Beschleunigung durch die Raketendüsen aufaddiert, ehe das y dann errechnet werden kann. Wenn man die Fähre durch Zünden der Steuerdüsen antreiben will, muß man also den Wert Delta y erhöhen, wenn man die Fähre vom Mond weg beschleunigen will. *Bild 2* zeigt das Struktogramm. Zunächst soll die Oberfläche des Mondes dargestellt werden, daher ist dies die erste Aktion im Programm. Dann werden die Anfangswerte für die Mondfähre bestimmt. Die Position und die y-Position der Fähre werden bestimmt, denn die Fähre soll irgendwo oberhalb der Mondoberfläche beginnen, so zumindest in unserem Spiel. Delta x kann man mit 0 belegen, wenn die Fähre keine Bewegung in x-Richtung durchführen soll. Man kann Delta x auch mit einer Konstanten belegen, wenn man der Fähre eine Anfangsgeschwindigkeit in x-Richtung mitgeben will um die Landung zu erschweren. Delta y wird mit dem Wert 0 belegt, die Fähre soll also im

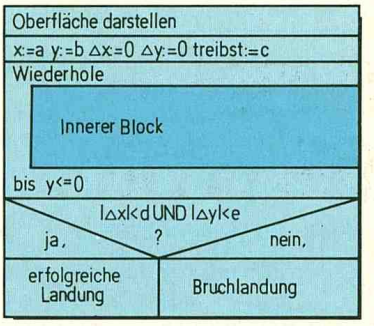


Bild 2. Das erste grobe Struktogramm

ersten Moment in y-Richtung stillstehen. Und der Treibstoffvorrat wird ebenfalls mit einer Konstanten belegt.

Das Landeprogramm wird geplant

Der innere Block, also der Spielblock des Programms soll dann so lange wiederholt werden, bis die Fähre eine y-Koordinate hat, die kleiner oder gleich 0 ist (bis also eine Landung erfolgte). Dann muß festgestellt werden, ob es eine Bruchlandung oder eine erfolgreiche Landung gewesen ist. Dies kann man zum Beispiel dadurch tun, daß man das letzte Delta x und das letzte Delta y als Betrag mit zwei Konstanten vergleicht. Denn wenn einer der genannten Beträge zu groß ist, landet die Fähre mit einer zu hohen Geschwindigkeit in x- oder y-Richtung. Eine zu hohe Landegeschwindigkeit führt zur Bruchlandung. Man könnte ferner auch noch die y-Koordinate prüfen, denn wenn deren Wert viel kleiner als 0 ist, so landet man unter der Mondoberfläche und das ist sicher nicht gut, auch wenn man dort mit einer geringen Landegeschwindigkeit aufsetzt. Wenn man eine Mondoberfläche gestaltet, die hohe Berge und Täler besitzt, so muß man die Abbruchbedingung des inneren Blocks komplizierter gestalten. Man muß dann die y-Koordinate in Abhängigkeit von der x-Koordinate betrachten. Wenn y kleiner gleich yMOND(x) ist, dann ist man schon gelandet. Doch dies nur als Anregung für eigene Gestaltungen des Spiels.

Die Flugphase

Nun zum inneren Block. *Bild 3* zeigt das Struktogramm. Dort wird zunächst die Fähre an der neuen x,y-Position ausgegeben und das alte Bild gelöscht. Dann wird abgefragt: Wenn der Treibstoffvorrat größer als 0 ist, so kann man die Fähre noch manövrieren; ohne Treibstoff geht nichts mehr, die Fähre fällt im freien Fall auf die Mondoberfläche zurück. Wenn also noch Treibstoff vorhanden ist, so kann man das Tastenfeld abfragen. Es sollen vier Steuertasten verwendet werden. Wenn man die Taste „Düse rechts" drückt, so wird Delta x um einen konstanten Wert erhöht. Man kann hier zum Beispiel den Wert 3 wählen. Die Größe des Wertes gibt wie gesagt die Beschleunigung an. Man könnte diesen Wert bei geringer werdendem Treibstoffvorrat zum Beispiel größer machen. Wenn man die Taste „Düse links" betätigt, so wird Delta x um den Wert 3 verringert. Bei der Taste „Rauf" wird Delta y um 3 erhöht. Damit kann man die Fähre von der Oberfläche weg beschleunigen, und bei „Runter" wird Delta y um 3 verringert. Nach dieser Abfrage wird Delta y noch zusätzlich um 1 verringert. Dies entspricht der Gravitation des Mondes. Dieser Wert muß kleiner sein als der Wert für die Beschleunigung mit den Düsen, sonst kann man die Gravitationskraft nicht durch den Schub der Düsen aufheben.
Die Gravitation sei übrigens bei uns unabhängig von der Höhe über der Mondoberfläche.
Nun erfolgt die neue Berechnung der x- und y-Koordinaten, indem die Delta-Werte zu den alten Koordinaten addiert werden. Jetzt wird noch der Treibstoffvorrat verringert. Schließlich noch eine Warteschleife, denn der Computer ist sehr schnell und die Landung sollte ja nur so schnell vorgeführt werden, daß man noch reagieren kann. Mit der Dauer der Schleife wird praktisch die Zeitdifferenz t2 – t1 = Delta t bestimmt, die oben schon verwendet wurde.

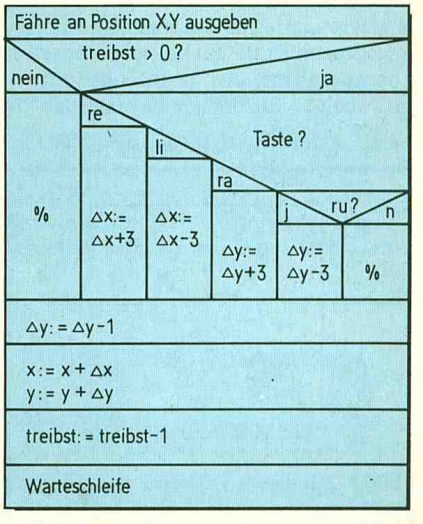


Bild 3. Eine Verfeinerung des inneren Blokkes aus Bild 2

Die Landefähre

Nun haben wir noch ein Problem, die Darstellung der Landefähre. Dazu gibt es im Grundprogramm eine Routine, die speziell für bewegte Objekte gedacht ist. Sie bildet den Figur-Befehl. Eine darzustellende Figur wird dazu in einzelne Vektoren zerlegt. So kann sie in codierter Form eingegeben werden. Der Figur-Befehl löscht eine solche Figur immer erst an der alten Position und baut sie an einer vorzugebenden neuen Position wieder auf. Das geht so schnell, daß sich eine Bewegung auf dem Bildschirm zeigt. *Bild 4* zeigt die Codierung der Vektoren. In *Bild 5* ist aus diesen Vektoren eine Mondlandefähre aufgebaut. Der Figur-Befehl benötigt neben den Vektorkennungen 0 bis 7 auch noch ein Ende-Zeichen sowie eine Möglichkeit, einen „Dunkelvektor" zu zeichnen. Als Zeichen für das Ende wurde die Zahl 10 gewählt. Mit dem Code 8 kann man den Schreibstift heben. Es werden dann nur noch unsichtbare Vektoren gezeichnet, was mit 9 wieder aufgehoben werden kann. *Tabelle 1* zeigt eine Zusammenstellung der Befehle. *Bild 6* zeigt ein kleines Testprogramm, das eine Figur an einer x,y-Position aufbaut.

Die Probezeichnung

Zunächst wird die Figur definiert, indem die Vektoren passend aufgezählt werden. Man gibt der Tabelle am besten einen Namen, hier z.B. den Namen „faehre". Dann folgt ein Assembler-Befehl, den wir noch nicht kennen: dc.b. dc bedeutet Define Constant, deutsch: definiere Konstante. Damit ist es möglich, direkt Zahlenwerte im Speicher abzulegen. Das „.b" bedeutet, daß Bytes abgelegt werden sollen, daß also eine angegebene Zahl 8 Bit belegt. Alternativ könnte man mit „.w" 16-Bit- oder mit „.1" auch 32-Bit-Daten ablegen. Doch

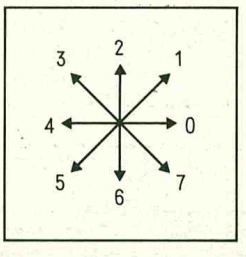


Bild 4. Mit diesen Vektoren, von welchen man beliebig viele Exemplare benutzen kann, kann eine Figur aufgebaut werden

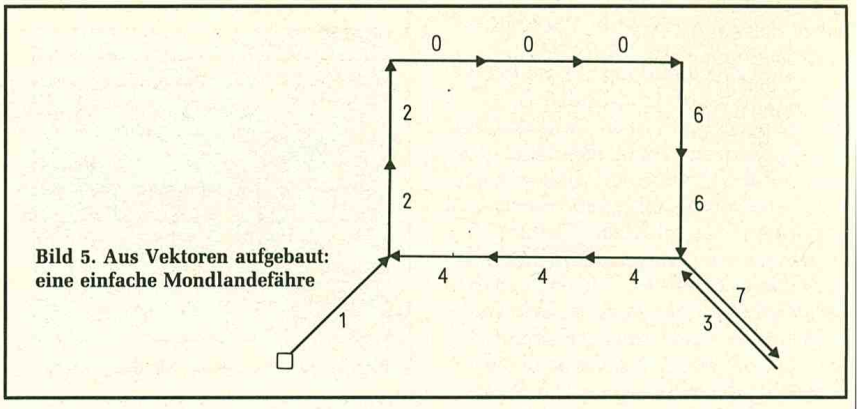


Bild 5. Aus Vektoren aufgebaut: eine einfache Mondlandefähre

der Figurbefehl kommt mit einer Byte-Größe aus, da ja nur die Zahlen 0 bis 10 verwendet werden. Nun werden die Vektoren einfach in codierter Form in die auf dc.b folgende Tabelle geschrieben. Wenn der Platz in einer Zeile einmal nicht reichen sollte, kann man jederzeit in der nächsten Zeile fortfahren, wenn man erneut den dc.b-Befehl voranstellt. Wenn die einzelnen Zahlenwerte durch Komma getrennt werden, so werden sie im Computer fortlaufend in aufeinanderfolgende Bytes eingetragen. Dann folgt ein neuer Assembler-Befehl: ds 0. ds bedeutet define storage, belege Speicherplatz. Mit ds.w 100 kann man ein Feld von 100 16-Bit-Worten reservieren, zum Beispiel als Arbeitsspeicher. ds 0 besitzt eine besondere Bedeutung.

Beim 68008- und 68000-Prozessor müssen Befehle immer auf einer geraden Speicheradresse beginnen, sonst erscheint eine Fehlermeldung (ADRESS ERROR) bei der Ausführung. Auch der Assembler prüft das und warnt rechtzeitig. Wenn man mit dc.b einzelne Bytes im Speicher ablegt, so hängt es von der Anzahl der Bytes ab, ob die nächste Speicherzelle auf einer geraden oder ungeraden Adresse liegt. Mit ds 0 wird nun der Assembler veranlaßt, automatisch ein zusätzliches Byte einzufügen, wenn eine ungerade Adresse vorliegt. Der nächste Befehl wird dann vom Assembler garantiert auf einer geraden Adresse abgelegt. Die Forderung wiederum, daß Befehle auf einer geraden Adresse liegen müssen, ist eine Eigen-

```
faehre: dc.b 1,2,2,0,0,0,6,6,7,3,4,4,4,10
 ds 0

start:
 lea faehre,a0   * Adresse der Fähre
 move #3,d0      * Groesse der Fähre
 move #100,d1    * X-Koordinate
 move #180,d2    * Y-Koordinate
 jsr @figur      * und ausführen
 rts

 end
```

Bild 6. Dieses Programm zeichnet die Landefähre

heit des Prozessors 68008. Sie kommt daher, daß ein Befehlswort bei diesem Prozessor immer mindestens 16 Bit groß ist und daß er vom 68000 abgeleitet ist. Der 68000 besitzt einen 16-Bit-Datenbus. Er kann daher einen Befehl von Wortbreite mit einem Zugriff direkt holen. Läge bei ihm ein Befehl auf einer ungeraden Adresse, so müßte er zwei Zugriffe durchführen. Dadurch wird bei ihm die Ausführungszeit größer. Beim 68008 wäre dies kein Problem, denn er besitzt einen 8-Bit-Datenbus. Aber man hat diese Konvention aus Kompatibilitätsgründen aufrechterhalten und so muß man beim Programmieren darauf achten, also immer ds 0 nach Assembler-Befehlen, die einzelne Bytes im Speicher ablegen, und alles geht gut.

Das Hauptprogramm

Das Hauptprogramm beginnt bei der Marke „start:". Danach wird zuerst der Befehl „lea faehre,a0" ausgeführt. Dieser Befehl bedeutet load effective adress, oder lade aktuelle Adresse. Das Ziel ist ein Adreßregister, nämlich A0. Die Wirkung besteht darin, daß die Adresse der Tabelle „faehre" in das Register A0 geladen wird. Man hätte hier auch „MOVE.A.L #faehre,a0" schreiben können, also einen Transportbefehl ins Adreßregister mit direktem Operanden verwenden können, doch sollte man hier besser den LEA-Befehl verwenden, der für diesen Zweck gedacht ist und der die Lesbarkeit eines Programms erhöht. Beim 68008 gibt es oft mehrere Befehle, mit denen man ein und dieselbe Aufgabe erfüllen kann, man lernt erst nach und nach die Feinheiten und Unterschiede kennen. Mit dem nachfolgenden „MOVE"-Befehl wird der Wert 3 in das Register D0 geschafft. Dort verlangt der Figur-Befehl den Vergrößerungsfaktor. Ein Vektor kann nämlich minimal einen Punkt lang sein. Will man längere Vektoren darstellen, so kann man dies einmal durch Aneinanderreihen von einzelnen Vektoren erreichen oder besser einen Vergrößerungsfaktor verwenden (wenn man nicht zusätzlich feine Details an anderen Stellen der Darstellung benötigt). Die Werte 1 bis 3 für den Vergrößerungsfaktor sind in der Ausführung besonders schnell, da der Grafikprozessor selbst die Vergrößerung übernimmt. Bei 4..n muß dies vom Grundprogramm mit Hilfe einer Schleife durchgerechnet werden. Die Ausführung ist dann nicht mehr so schnell, bei großen Figuren kann es daher zum Flimmern kommen, weil man den Schreibvorgang beobachten kann. Wenn man übrigens den Wert 0 als Vergrößerung angibt, so wird nur die alte Figur gelöscht, aber keine neue mehr gezeichnet. So kann man eine Figur auch wieder ganz löschen.
Im Register D1.W wird die x-Koordinate (0 bis 511) und im Register D2.W die y-Koordinate (0 bis 255) an den Grafikprozessor übergeben. Wählt man einen Wert, der außerhalb dieses Bereiches liegt, so wird die Figur teilweise oder ganz verschwinden. Dabei ist die absolute Grenze 4096, dann erscheint die Figur von der anderen Seite her wieder im Bild.

Tabelle 1.
Der Figur-Befehl ist komfortabel

1,2,3...7	Richtungscode
8	Schreibstift heben
9	Schreibstift senken
10	Ende der Tabelle

Der Figur-Befehl merkt sich zum einen die alten Koordinaten der Figur, zum anderen auch die Adresse und Größe der Figur. Dadurch kann beim erneuten Aufruf mit veränderten Werten die alte Figur zunächst gelöscht und dann die neue Figur gezeichnet werden. Will man eine Figur einfrieren, zum Beispiel weil man zusätzlich eine Zweite ins Bild bringen will, so kann man dies durch den Aufruf von „SETFIG" tun. Dann wird die alte Figur beim erneuten Aufruf nicht mehr gelöscht, sondern die neue Figur an neuer Stelle dargestellt.
Bild 7 zeigt das Ergebnis eines Programmlaufes nach Bild 6.

Die Tasten an IOE

Das Tastenfeld kann man, wie schon früh gezeigt, an die IOE-Baugruppe anschließen. *Bild 8* zeigt die Belegung der Tasten. Bit 0 des Ports 30h (I/O 0) bilden dabei die Taste für Rechts usw.

Das komplette Programm

Bild 9 zeigt das Listing des kompletten Programms „Mondlandung". Dabei tauchen eine Reihe von neuen Befehlen auf. Mit EQU wird im „Vereinbarungsbaustein" zunächst die Portadresse für das Tastenfeld definiert. Dann werden sogenannte Bitmasken definiert. Damit ist es später möglich, die Tastenabfrage einfach zu gestalten. Mit dem „DC.B"-Befehl kann man nicht nur Zahlen, sondern auch Texte ablegen, wenn sie zwischen zwei einfachen Anführungszeichen liegen. So werden hier die beiden Texte „Gut gelandet" und „Bruchlandung" als ASCII-Text im Speicher abgelegt. Mit „,0" wird noch der Code 0 im Speicher hinter dem Text abgelegt, um das Ende des jeweiligen Textes für die Ausgaberoutine zu kennzeichnen. Die Definition der Tabelle „Faehre" läuft wie besprochen. Dann folgt der Befehl „DS 0", mit dem die nächste gerade Adresse eingestellt wird. Im Listing sieht man, daß hier ds 0 nicht nötig gewesen wäre, denn die Adresse „A02E" ist die nächste Adresse und sie ist gerade. Wenn man aber eigene Fährenmodelle codieren möchte, so kann man möglicherweise „DS 0" gut gebrauchen, weil damit keine Probleme bei der Befehlsablage auftreten.
Jetzt folgt ein Block, der mit „Variablenvereinbarungen" beschriftet ist. Dort stehen alle Variablen des Programms, also Speicherzellen, deren Inhalt im Laufe des Programms verändert werden. Zum Beispiel die Variable für die x-Koordinate, die mit „X:" markiert ist. Mit „dc.w 0" wird ein 16-Bit-Wort reserviert und der Wert 0 als Startwert abgelegt. Dann folgen die Variablen Y,DX,DY und TREIBST mit der gleichen Bedeutung, wie im Struktogramm.

Bild 7. In realer Größe: die Landefähre

Die Unterprogramme

MONDFLAECHE ist der erste Unterprogrammbaustein. Dort wird die Mondoberfläche erzeugt. Sie soll bei unserem Mond nur mit gleichförmigen Kratern versehen sein, so daß man überall lan-

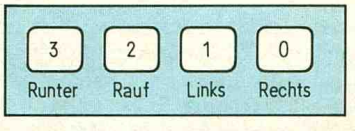


Bild 8. Diese Tasten müssen an die IOE-Karte angeschlossen werden

den kann. Dazu wird das Unterprogramm KRATER häufig fünfzigmal aufgerufen. Mit dem Befehl SET kann man die Startposition der Schildkröte festlegen. X und Y (D1 und D2) werden mit 0 belegt und D3 mit 90, also blickt die Schildkröte nach oben und befindet sich links unten am Bildschirmrand. Mit HIDE wird verhindert, daß man die Schildkröte selbst sieht, denn auf dem Mond gibt es keine Schildkröten. Mit einem DBRA-Befehl wird eine Schleife realisiert, die mit JSR KRATER das Unterprogramm Krater aufruft: Mit DREHE und SCHREITE wird in dem Unterprogramm KRATER ein einfacher Hügel gezeichnet.

Dann folgt im Programmbaustein das eigentliche Hauptprogramm, das bei der Marke „START:" beginnt. Mit BSR MONDFLAECHE wird zunächst, wie im Struktogramm vorgeschrieben, die Mondoberfläche ausgegeben. Dabei wird hier einmal ein BSR-Befehl zum Unterprogrammaufrufen verwendet. Mit JSR könnte man das auch tun. BSR ist aber kürzer, er adressiert relativ. Das heißt, daß in dem vom Assembler erzeugten Maschinencode nur die Differenz zwischen der Position des Aufrufs und des Ziels vermerkt wird. Der JSR-Befehl (siehe Unterprogramm MONDFLAECHE) verwendet dagegen eine absolute Adresse. Wenn man später einmal ein Programm verschieben will, also auf einer anderen Adresse ablaufen lassen möchte, als es mit der ORG-Anweisung ursprünglich vorgesehen war, so müssen alle absoluten Adressen geändert werden. Im Gegensatz dazu bleiben die relativen Anweisungen gültig. Wenn man also einmal verschiebbare Programme konstruieren möchte, so ist die Verwendung von Befehlen mit relativer Adressierung sinnvoll. Ein anderer Vorteil des BSR-Befehls ist, daß man einen ganz kurzen Maschinenbefehl erhält, wenn man ein „.S" an den Befehl anhängt, also „BSR.S" schreibt. Denn die relative Distanz wird dann mit in den 16 Bit breiten Befehl eingebaut. Allerdings darf das

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1
009C00                  ;*********************************
009C00                  ; MONDLANDEPROGRAMM VERSION 1.0 ;
009C00                  ; VOM 13.12.1983                ;
009C00                  ;*********************************
009C00
00A000                   ORG $A000
00A000
00A000                  ; ***** VEREINBARUNGS-BAUSTEIN *****
00A000
= FFFFFF30              TASTEN EQU $FFFFFF30
= 00000001              RECHTS EQU 1    ; BIT MASKEN
= 00000002              LINKS  EQU 2
= 00000004              RAUF   EQU 4
= 00000008              RUNTER EQU 8
00A000
00A000                  ERFOLGTEXT:
00A000  4775742067656C   DC.B 'Gut gelandet',0
00A007  616E64657400
00A00D
00A00D                  BRUCHTEXT:
00A00D  42727563686C61   DC.B 'Bruchlandung',0
00A014  6E64756E6700
00A01A
00A01A                  FAEHRE:
00A01A  08 02 09 00 00   DC.B 8,2,9,0,0,7,3,2,2,4,4,4,4
00A01F  07 03 02 02 04
00A024  04 04 04
00A027  06 06 05 01 00   DC.B 6,6,5,1,0,0,10
00A02C  00 0A
00A02E
00A02E                   DS 0 ; ANGLEICH AUF WORTGRENZE
00A02E
00A02E                  ; VARIABLENVEREINBARUNGEN
00A02E
00A02E  0000            X: DC.W  0
00A030  0000            Y: DC.W  0
00A032  0000            DX: DC.W  0
00A034  0000            DY: DC.W  0
00A036  0000            TREIBST: DC.W  0
00A038
00A038                  ; ***** UNTERPROGRAMM-BAUSTEIN *****
00A038
00A038                  MONDFLAECHE:
00A038  323C 0000        MOVE #0,D1
00A03C  343C 0000        MOVE #0,D2
00A040  363C 005A        MOVE #90,D3
00A044  4EB9 0000131A    JSR @SET       ;SETZE SCHILDKROETE
00A04A  4EB9 00001112    JSR @HIDE      ;ABER NICHT ZEIGEN
00A050  383C 0031        MOVE #50-1,D4     ;ANZAHL KRATER
00A054                  MLP1:
00A054  4EB9 0000A060    JSR KRATER
00A05A  51CC FFF8        DBRA D4,MLP1
00A05E  4E75             RTS
00A060
00A060                  KRATER:
00A060  303C FFD3        MOVE  #-45,D0
00A064  4EB9 000012F6    JSR @DREHE
00A06A  303C 000A        MOVE #10,D0
00A06E  4EB9 00001288    JSR @SCHREITE
00A074  303C FFD3        MOVE #-45,D0
00A078  4EB9 000012F6    JSR @DREHE
00A07E  303C 0006        MOVE #6,D0
00A082  4EB9 00001288    JSR @SCHREITE
00A088  303C FFD3        MOVE #-45,D0
00A08C  4EB9 000012F6    JSR @DREHE
00A092  303C 000A        MOVE #10,D0
00A096  4EB9 00001288    JSR @SCHREITE
00A09C  303C FF1F        MOVE #-45-180,D0
00A0A0  4EB9 000012F6    JSR @DREHE
00A0A6  4E75             RTS
00A0A8
00A0A8                  ; ***** PROGRAMM-BAUSTEIN *****
00A0A8
00A0A8                  START:
00A0A8  6100 FF8E        BSR MONDFLAECHE
00A0AC  33FC 0064        MOVE #100,X
00A0B0  0000A02E
00A0B4  33FC 00C8        MOVE #200,Y
00A0B8  0000A030
00A0BC  33FC 0003        MOVE #3,DX
00A0C0  0000A032
00A0C4  33FC 0000        MOVE #0,DY
00A0C8  0000A034
00A0CC  33FC 012C        MOVE #300,TREIBST
00A0D0  0000A036
00A0D4                  WDH1:
00A0D4  207C 0000A01A    MOVEA.L #FAEHRE,A0
00A0DA  303C 0003        MOVE #$3,D0        ; VERGROESSERUNG=3
00A0DE  3239 0000A02E    MOVE X,D1
00A0E4  3439 0000A030    MOVE Y,D2
00A0EA  4EB9 0000326C    JSR @FIGUR
00A0F0  4A79 0000A036    TST TREIBST
00A0F6  6F00 0052        BLE NOTEST         ; KEIN TREIBSTOFF ?
00A0FA  1039 FFFFFF30    MOVE.B TASTEN,D0
00A100  0200 0001        AND.B #RECHTS,D0
00A104  6608             BNE.S TST1
00A106  5679 0000A032    ADDQ #3,DX
00A10C  603C             BRA.S NOTEST
00A10E
00A10E                  TST1:
00A10E  1039 FFFFFF30    MOVE.B TASTEN,D0
00A114  0200 0002        AND.B #LINKS,D0
00A118  6600 000A        BNE TST2
00A11C  5779 0000A032    SUBQ #3,DX
00A122  6026             BRA.S NOTEST
00A124
00A124                  TST2:
00A124  1039 FFFFFF30    MOVE.B TASTEN,D0
00A12A  0200 0004        AND.B #RAUF,D0
00A12E  6608             BNE.S TST3
00A130  5679 0000A034    ADDQ #3,DY
00A136  6012             BRA.S NOTEST
00A138
00A138                  TST3:
```

Bild 9. Das Programm zur Mondlandung ist lang

Sprungziel dann nicht allzuweit vom Aufruf entfernt sein, maximal etwa 126 Bytes in beiden Richtungen. Sonst paßt die Distanz nicht mehr in ein Byte und man muß die längere Form angeben. Da kann das Sprungziel schon ca. ±32765 Bytes entfernt sein. Wenn man noch größere Sprünge machen möchte, so muß man den JSR-Befehl verwenden. So müssen auch Grundprogrammbefehle mit JSR aufgerufen werden, denn da kann es leicht passieren, daß der Abstand größer als 32 KByte wird. Außerdem reagiert der Einzelschritt-Modus des Grundprogrammes unterschiedlich auf die beiden Befehle. Beim JSR-Befehl wird das Unterprogramm als Ganzes ausgeführt. Beim BSR-Befehl wird dagegen das gerufene Unterprogramm schrittweise durchlaufen.
Man kann die unterschiedliche Wirkung zwischen JSR und BSR also zu Testzwecken gezielt einsetzen.
Zurück zum Programm. Im Struktogramm stand nun die Initialisierung der Variablen. Dies wird als nächstes getan. Die Fähre soll bei der Position X=100 und Y=200 starten. In X-Richtung soll die Anfangsgeschwindigkeit 3 verwendet werden, um den Spielablauf zu erschweren. Gemäß Struktogramm könnte man auch den Wert 0 laden. DY wird mit 0 belegt, die Fähre hat also in Y-Richtung zunächst keine Geschwindigkeit.

```
00A138  1039 FFFFFF30    MOVE.B TASTEN,D0
00A13E  0200 0008        AND.B #RUNTER,D0
00A142  6606             BNE.S NOTEST
00A144  5779 0000A034    SUBQ #3,DY
00A14A
00A14A                   NOTEST:
00A14A  5379 0000A034    SUBQ #1,DY            ;GRAVITATION
00A150  3639 0000A032    MOVE DX,D3            ; X:=X+DX
00A156  D779 0000A02E    ADD D3,X
00A15C  3639 0000A034    MOVE DY,D3            ; Y:=Y+DY
00A162  D779 0000A030    ADD D3,Y
00A168  5379 0000A036    SUBQ #1,TREIBST       ; TREIBSTOFF VERINGERN
00A16E
00A16E  323C 0004        MOVE #5-1,D1          ; WARTESCHLEIFE
00A172                   WARTE20MS:
00A172  3E3C 001C        MOVE #!SYNC,D7
00A176  4E41             TRAP #1
00A178  6700 FFF8        BEQ WARTE20MS
00A17C  51C9 FFF4        DBRA D1,WARTE20MS
00A180
00A180  0C79 0005        CMP #5,Y              ; HOEHE DER OBERFLAECHE UEBER 0
00A184  0000A030
00A188  6E00 FF4A        BGT WDH1              ; BERUECKSICHTIGEN
00A18C                                         ;
00A18C  303C 0003        MOVE #$3,D0           ; FAEHRE AM SCHLUSS NOCHMALS ZEIGEN
00A190  3239 0000A02E    MOVE X,D1
00A196  3439 0000A030    MOVE Y,D2
00A19C  4EB9 0000326C    JSR @FIGUR            ; FIGUR AUFRUFEN
00A1A2  3039 0000A032    MOVE DX,D0
00A1A8  6A02             BPL.S FF1
00A1AA  4440             NEG D0
00A1AC                   FF1:                  ; !DX! < 2
00A1AC  0C40 0002        CMP #2,D0
00A1B0  6C18             BGE.S BRUCH
00A1B2  3039 0000A034    MOVE DY,D0
00A1B8  6A02             BPL.S FF2
00A1BA  4440             NEG D0
00A1BC                   FF2:                  ; !DY! < 4
00A1BC  0C40 0004        CMP #4,D0
00A1C0  6C08             BGE.S BRUCH
00A1C2  41F9 0000A000    LEA ERFOLGTEXT,A0
00A1C8  6006             BRA.S ENDE
00A1CA
00A1CA                   BRUCH:
00A1CA  41F9 0000A00D    LEA BRUCHTEXT,A0
00A1D0                   ENDE:
00A1D0  303C 0033        MOVE #$33,D0          ; SCHRIFTGROESSE
00A1D4  323C 0000        MOVE #0,D1            ; X-POSITION
00A1D8  343C 0082        MOVE #130,D2          ; Y-POSITION
00A1DC  4EB9 00001386    JSR @WRITE
00A1E2  4EB9 00000A00    JSR @CI               ; WARTEN AUF TASTE
00A1E8  4EB9 00000DA0    JSR @CLR              ; LOESCHEN
00A1EE  6000 FEB8        BRA START             ; UNBEGRENZT
00A1F2
00A1F2                   ; ***********
00A1F2
00A1F2                   END

008C34  Ende-Symboltabelle
```

Der Treibstoffvorrat wird mit 300 Einheiten festgesetzt. Dann folgt die Marke für eine Wiederholschleife „WDH1:". Dort beginnt der eigentliche Spielblock, der „Innere Block".

Zunächst wird die Fähre an der X,Y-Position ausgegeben. Mit „MOVE X,D1" wird z.B. die X-Koordinate in das Register D1 geschafft, so wie es der FIGUR-Befehl zur Bearbeitung benötigt.
Dann folgt ein neuer Befehl „TST TREIBST". Damit werden die Bedingungsflags des 68008 gesetzt. Es wird geprüft, ob der Wert 0 ist, wenn ja, so wird das Z-Flag (Zero, Null) gesetzt, dann wird geprüft, ob der Wert kleiner Null ist und wenn ja, so wird das M-Flag (Minus) gesetzt. All dies macht der TST-Befehl automatisch, ohne andere Register zu verändern. Nun kann man mit BLE (branch less or equal) die Entscheidung „TREIBST <=0" durchführen. Wenn der Inhalt der Variablen TREIBST kleiner oder gleich 0 war, so erfolgt ein Sprung zur Marke NOTEST, denn dann ist der Absturz nahe und das Tastenfeld ohne Wirkung. Mit Treibstoff läuft das Programm weiter. Es wird dann zunächst der Port TASTEN mit „MOVE.B TASTEN,D0" in das Register D0 eingelesen. Dann wird eine Und-Verknüpfung durchgeführt. Dabei wird der Wert der Konstanten RECHTS mit D0 verknüpft. RECHTS hatte den Wert 1, daher ist danach das Register D0.B =0, wenn das Bit 0 des Registers D0.B vorher 0 war, sonst ungleich 0. Das Ergebnis ist unabhängig von der Stellung der restlichen Bits, die ausgeblendet wurden. Damit wird also die Taste 0 abgefragt. Mit dem Sprungbefehl BNE TST1 springt das Programm weiter, wenn die Taste RECHTS nicht betätigt wurde. Denn wenn die Taste gedrückt wird, so gelangt 0V an Bit 0 des Ports der IOE-Baugruppe und 0V entspricht der logischen 0. Unbetätigte Tasten dagegen erzeugen eine 1 am Eingang (auch wenn der Eingang offen bleibt, das ist bei TTL-ICs so).

Im Programm wird, wenn die Taste RECHTS gedrückt wurde, mit ADDQ #3,DX der Wert 3 zu delta x addiert. Das ist die Beschleunigung. Danach erfolgt ein Sprung zur Marke NOTEST, denn damit ist der gesamte Test beendet.
Bei TST1: wird nach dem Einlesen des Ports Bit 1 durch eine UND-Verknüpfung mit dem Wert 2 erreicht, daß das Bit 1 isoliert wird. Dann wird das Bit 1 abgefragt. Ist das Ergebnis der UND-Verknüpfung 0, so wurde die Taste gedrückt und der Wert 3 wird mit SUBQ #3,DX subtrahiert.

Quick

Das Q hinter ADD und SUB bedeutet Quick, also schnell. Dieser Zusatz ist nur bei insgesamt drei 68008-Befehlen möglich (ADDQ SUBQ MOVEQ). Er bewirkt, daß der zu addierende Wert direkt in den Maschinencode eingearbeitet wird. Daher belegt der Befehl weniger Bytes im Hauptspeicher und ist bei der Ausführung sehr schnell. Allerdings ist der Zahlenbereich begrenzt, den er verarbeiten kann. Bei ADDQ und SUBQ sind nur Werte zwischen 1 und 8 erlaubt, bei MOVEQ ist der Wertebereich von –128 bis +127 erlaubt. Als Ziel sind nur die Datenregister D0 bis D7 erlaubt und es werden alle 32 Bit belegt. Ist die Zahl kleiner 0 (oder größer 128), so wird sie als Zweierkomplement aufgefaßt und das Vorzeichen wird auf alle restlichen Bits übertragen.
Zurück zum Programm. Nach dem Abfragen der Tasten folgt die Marke NOTEST.

Jetzt wird gerechnet

Dort wird nun der Rest der Schleife durchgearbeitet. Der Wert von DY wird verringert, und die neuen Koordinaten werden berechnet. Auch der Treibstoffvorrat wird ständig verringert. Man könnte hier das Programm so verändern, daß der Treibstoffvorrat nur dann verringert wird, wenn auch die Düsen in Betrieb sind. Hier sind der Phantasie keine Grenzen gesetzt.
Nun folgt die Marke WARTE20MS:. Dort wird das Unterprogramm SYNC aufgerufen, diesmal zur Abwechslung mit einem TRAP-Befehl, genauso würde es aber auch mit JSR @SYNC gehen. Der Befehl SYNC setzt nur ein Flag. Das Zero-Flag (Null-Flag) ist gesetzt, wenn beim Durchlaufen von Sync kein Synchronsignal am GDP anliegt (das Synchronsignal ist die Vertikalsynchronisation des GDP). Dieser Impuls erscheint sehr genau alle 20 Millisekunden. Ist der Wert ungleich 0, so ist der Puls aufgetreten. Die Sync-Abfrageroutine bewirkt noch, daß das Synchronsignal nach einer Abfrage zurückgesetzt wird, damit es erst wieder nach genau 20 ms auf 1 geht. Die Schleife wird insgesamt 5mal durchlaufen, so daß der Innere Block nun 100 ms wartet, bevor die Ausführung weiter geht. Nun wird die Höhe Y mit dem Wert verglichen. Dies geschieht mit dem Befehl CMP #5,Y. Die Mondoberfläche ist nicht ganz eben, sie ragt 5 Einheiten empor. Damit stößt aber die Mondfähre abweichend vom Struktogramm schon bei einer Koordinate größer 0 auf Grund.

Bild 10. Eine gute...

Bild 11. ...und eine schlechte Landung

Wenn man eine ganz unregelmäßige Struktur der Oberfläche einbaut, so muß man hier die Abfrage in Abhängigkeit von der X-Koordinate durchführen. Mit dem CMP-Befehl wird eine Subtraktion durchgeführt. Und zwar wird von dem Wert der Y-Koordinate der Wert 5 probeweise abgezogen. Der WERT CMP setzt nämlich nur die Flags und verändert sonst keine Registerinhalte. Mit einem bedingten Sprung kann man anschließend die Entscheidung treffen „gelandet" oder „nicht gelandet". Mit BGT, das bedeutet „Branch Greater", wird der Sprung zur Marke WDH1 durchgeführt, wenn Y größer als 5 war. Dann schwebt die Mondlandefähre noch über Grund und die Schleife muß wiederholt werden. Im anderen Fall muß man nun prüfen, ob eine Bruchlandung vorliegt oder nicht. Die Ausgabe des Textes „Bruchlandung" oder „gut gelandet" erfolgt mit dem Unterprogramm WRITE, das im Grundprogramm steht. Dabei erwartet WRITE im Register A0 die Adresse des auszugebenden Textes. Entsprechend wird in Abhängigkeit von den Werten DX und DY das Register A0 entweder mit der Adresse der Marke „ERFOLGTEXT" oder „BRUCHTEXT" geladen. Bei ENDE: erfolgt die Ausgabe. Dazu wird in das Register D0 der Code für die Schriftgröße geladen. Der Wert berechnet sich wie folgt:
Wert := textbreite * 16 + texthoehe
Dabei ist „textbreite" und „texthoehe" im Bereich 1 bis 15. Wenn man den Wert 16 erhalten will, so muß man dafür 0 einsetzen. Im Register D1 steht dann die X-Koordinate des ersten Buchstabens und im Register D2 die Y-Koordinate des ersten Buchstabens. Mit JSR @WRITE wird der Text auf den Bildschirm ausgegeben. Nun folgt der Aufruf von CI, und der Rechner wartet, bis man eine Taste drückt, dann wird mit JSR @CLR das Bild gelöscht und das Programm beginnt von neuem. *Bild 10* und *Bild 11* zeigen das Bild nach einer Landung.

Aufgaben:

1. Ergänzen Sie das Struktogramm, so daß es auch die Wiederholung mit der Tasteneingabe am Schluß wiedergibt.

2. Erweitern Sie das Programm, so daß es dem Pfichtenheft entspricht und zum Beispiel die Forderung „Ebener Landeplatz" erfüllt oder indem Sie z. B. ein Hochplateau einbauen, auf dem man landen muß.

3. Wie kann man die Geschwindigkeit des Spielablaufs beeinflussen?

4. Was passiert, wenn man den Warte-Block wegläßt (ausprobieren, indem man die Sprünge BEQ WARTE20MS und DBRA D1,WARTE20MS herauslöscht)?

5. Wie kann man die Größe der Landefähre durch Ändern nur eines Wertes variieren?

6. Überlegen Sie, wie man das Spiel erweitern kann, so daß bei Unterschreiten einer Mindesthöhe ein vergrößertes Bild erscheint und umgekehrt.

7. *Bild 12* zeigt das Listing eines Programms, das unterschiedliche Fährenmodelle nacheinander ausgibt. Konstruieren Sie das dazugehörige Struktogramm des Ausgabeprogramms. (COS bildet den Cosinus eines Wertes im D0.W-Register. Dabei wird der Wert mit 256 multipliziert und vor der Ausgabe gerundet. Der Wert erscheint im Register D0.W im Zweierkomplement.) *Bild 13* zeigt das Programmergebnis nach der Landung.

8. Wie kann man mehrere Fähren gleichzeitig laden lassen, die durch unterschiedliche Spieler gesteuert werden? Konstruieren Sie ein Programm dazu.

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

009C00                          * FAEHRENMODELLE
009C00
00A000                           ORG    $A000
00A000
00A000  00 01 01 00 00  FAEHRE: DC.B 0,1,1,0,0,0,0,0,7,7,0,4,3,3
00A005  00 00 00 07 07
00A00A  00 04 03 03
00A00E  02 04 04 04 04   DC.B 2,4,4,4,4,4,6,2,0
00A013  04 06 02 00
00A017  03 02 02 02 01   DC.B 3,2,2,2,1,0,0,0,7,6,6,6,5,10
00A01C  00 00 00 07 06
00A021  06 06 05 0A
00A025
00A025
00A025
00A025                  FAEHRE2:
00A025  00 01 01 00 00   DC.B 0,1,1,0,0,0,0,0,0,0,0,0
00A02A  00 00 00 00 00
00A02F  00 00
00A031  07 07 00 04 03   DC.B 7,7,0,4,3,3
00A036  03
00A037  02 02 02 02 03   DC.B 2,2,2,2,3,4,4,4,4,4,4,4
00A03C  04 04 04 04 04
00A041  04 04
00A043  05 06 06 06 06   DC.B 5,6,6,6,6,10
00A048  0A
00A049
00A049
00A049                  FAEHRE3:
00A049  00 01 01 01 03   DC.B 0,1,1,1,3,3,2,2,1,1,0,0,7,7,6,6,5,5
00A04E  03 02 02 01 01
00A053  00 00 07 07 06
00A058  06 05 05
00A05B  04 04 05 00 00   DC.B 4,4,5,0,0,6,6,4,0,0,4,2,2,0,0
00A060  06 06 04 00 00
00A065  04 02 02 00 00
00A06A  07 07 00 04 03   DC.B 7,7,0,4,3,3,3,10
00A06F  03 03 0A
00A072
00A072                  FAEHRE4:
00A072  00 01 01 01 02   DC.B 0,1,1,1,2,3,4,4,2,2,1,1,2,2,1
00A077  03 04 04 02 02
00A07C  01 01 02 02 01
00A081  00 00 00 00 07   DC.B 0,0,0,0,7,6,6,7,7,6,6,4,4,5,6
00A086  06 06 07 07 06
00A08B  06 04 04 05 06
00A090  07 07 07 00 04   DC.B 7,7,7,0,4,3,3,3,4,4,4,4,0,2,2
00A095  03 03 03 04 04
00A09A  04 04 00 02 02
00A09F  00 00 06 06 07   DC.B 0,0,6,6,7,7,0,4,3,3,4,4,5,5
00A0A4  07 00 04 03 03
00A0A9  04 04 05 05
00A0AD  04 0A            DC.B 4,10
00A0AF
00A0AF                  FAEHRE5:
00A0AF  08 02 09         DC.B 8,2,9
00A0B2  06 04 04 04 03   DC.B 6,4,4,4,3,2,2,2,2,7,7,6,6,0,0,4,4,3,3,4,4,6,7,4,4,4,4
00A0B7  02 02 02 02 07
00A0BC  07 06 06 00 00
00A0C1  04 04 03 03 04
00A0C6  04 06 07 04 04
00A0CB  04 04
00A0CD  01 02 00 00 04   DC.B 1,2,0,0,4,4,4,4,6,6,5,4,4,4,2,0,0,2,2,1,1,6,6,5,5,1,1
00A0D2  04 04 04 06 06
00A0D7  05 04 04 04 02
00A0DC  00 00 02 02 01
00A0E1  01 06 06 05 05
00A0E6  01 01
00A0E8  02 02 00 00 00   DC.B 2,2,0,0,0,0,0,0,4,4,1,1,2,2,3,3,4,4,5,5,6,6,7,7,10
00A0ED  00 00 00 04 04
00A0F2  01 01 02 02 03
00A0F7  03 04 04 05 05
00A0FC  06 06 07 07 0A
00A101
00A101
00A101                  FAEHRE6:
00A101  02 02 02 04 04   DC.B 2,2,2,4,4,4,3,4,3,4,3,2,1,0,1,0,1,0
00A106  04 03 04 03 04
00A10B  03 02 01 00 01
00A110  00 01 00
00A113  01 00 00 00 07   DC.B 1,0,0,0,07,4,4,4,4,4,4,0,0,0,0,0,0,0,7,0,7,0,7,6
00A118  04 04 04 04 04
00A11D  04 00 00 00 00
00A122 00 00 00 07 00
00A127 07 00 07 06
00A12B 02 04 04 04 04  DC.B 2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4
00A130 04 04 04 04 04
00A135 04 04 04 04 04
00A13A 04 04 04
00A13D 06 00 00 00 00  DC.B 6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
00A142 00 00 00 00 00
00A147 00 00 00 00 00
00A14C 00 00 00
00A14F 05 04 05 04 05  DC.B 5,4,5,4,5,4,4,4,4,4,0,0,6,6,6,10
00A154  04 04 04 04 04
00A159  00 00 06 06 06
00A15E  0A
00A15F
00A15F  00               DS 0
00A160
00A160                  START:
00A160  4EB9 00000A00    JSR @CI
00A166  363C 005A        MOVE #90,D3
00A16A                  STLP:
00A16A  41F9 0000A000    LEA FAEHRE,A0
00A170  3003             MOVE D3,D0
00A172  4EB9 00001030    JSR @COS
00A178  4440             NEG D0
00A17A  0640 0100        ADD #256,D0
00A17E  3400             MOVE D0,D2
00A180  303C 0003        MOVE #3,D0
00A184  323C 0002        MOVE #2,D1
00A188  4EB9 0000326C    JSR @FIGUR
00A18E                  ST1:
00A18E  4EB9 00000E38    JSR @SYNC
00A194  6700 FFF8        BEQ ST1
00A198  51CB FFD0        DBRA D3,STLP
00A19C  4EB9 00003266    JSR @SETFIG
00A1A2  363C 0064        MOVE #100,D3
00A1A6                  STLPB:
00A1A6  41F9 0000A025    LEA FAEHRE2,A0
00A1AC  6100 00F0        BSR SINCOS
00A1B0  303C 0003        MOVE #3,D0
00A1B4  323C 0046        MOVE #70,D1
00A1B8  4EB9 0000326C    JSR @FIGUR
00A1BE
00A1BE                  ST2:
00A1BE  4EB9 00000E38    JSR @SYNC
00A1C4  6700 FFF8        BEQ ST2
00A1C8  51CB FFDC        DBRA D3,STLPB
00A1CC  4EB9 00003266    JSR @SETFIG
00A1D2  363C 0064        MOVE #100,D3
00A1D6                  STLPD:
00A1D6  41F9 0000A049    LEA FAEHRE3,A0
00A1DC  6100 00C0        BSR SINCOS
00A1E0  303C 0003        MOVE #3,D0
00A1E4  323C 008C        MOVE #140,D1
00A1E8  4EB9 0000326C    JSR @FIGUR
00A1EE                  ST3:
00A1EE  4EB9 00000E38    JSR @SYNC
00A1F4  6700 FFF8        BEQ ST3
00A1F8  51CB FFDC        DBRA D3,STLPD
00A1FC  4EB9 00003266    JSR @SETFIG
00A202  363C 0064        MOVE #100,D3
00A206                  STLPE:
00A206  41F9 0000A072    LEA FAEHRE4,A0
00A20C  6100 0090        BSR SINCOS
00A210  303C 0003        MOVE #3,D0
00A214  323C 00D2        MOVE #210,D1
00A218  4EB9 0000326C    JSR @FIGUR
00A21E                  ST4:
00A21E  4EB9 00000E38    JSR @SYNC
00A224  6700 FFF8        BEQ ST4
00A228  51CB FFDC        DBRA D3,STLPE
00A22C  4EB9 00003266    JSR @SETFIG
00A232  363C 0064        MOVE #100,D3
00A236                  STLPF:
00A236  41F9 0000A0AF    LEA FAEHRE5,A0
```

Bild 12. Dieses Programm zeichnet verschiedene Fährenmodelle

FIGUR

Abschließend eine Besonderheit des Figurbefehls. Man kann damit auch in sich bewegte Figuren darstellen, die im Verlauf der Bewegung über den Bildschirm eine Eigenbewegung haben, wie z. B. der „Packman" in *Bild 14*. Dazu muß man für jede Bewegungsphase eine Figur verwenden. Da sich der Figur-Befehl auch die Adresse der Figur merkt, kann man durch Umschalten der Adresse diese innere Bewegung einfach erzeugen. *Bild 15* zeigt ein Programmbeispiel, mit dem die „Packman"-Figur schnappend über den Bildschirm bewegt wird.

1. Konstruieren Sie das dazugehörige Struktogramm.

2. Wieso wechselt Packman nicht bei jedem Schritt die Phase?

3. Was passiert, wenn man anstelle AND #8,D0 AND #16,D0 schreibt (Adresse $9C54)?

4. Verwenden Sie vier Bewegungsphasen für eine sanftere Bewegung und konstruieren Sie das dazugehörige Programm.

```
00A23C  6100 0060          BSR SINCOS
00A240  303C 0003          MOVE #3,D0
00A244  323C 0154          MOVE #340,D1
00A248  4EB9 0000326C      JSR @FIGUR
00A24E                     ST5:
00A24E  4EB9 00000E38      JSR @SYNC
00A254  6700 FFF8          BEQ ST5
00A258  51CB FFDC          DBRA D3,STLPF
00A25C  4EB9 00003266      JSR @SETFIG
00A262  363C 0064          MOVE #100,D3
00A266                     STLPG:
00A266  41F9 0000A101      LEA FAEHRE6,A0
00A26C  6100 0030          BSR SINCOS
00A270  303C 0003          MOVE #3,D0
00A274  323C 019A          MOVE #410,D1
00A278  4EB9 0000326C      JSR @FIGUR
00A27E                     ST6:
00A27E  4EB9 00000E38      JSR @SYNC
00A284  6700 FFF8          BEQ ST6
00A288  51CB FFDC          DBRA D3,STLPG
00A28C  4EB9 00000A00      JSR @CI
00A292  4EB9 00000DA0      JSR @CLR
00A298  6000 FEC6          BRA START
00A29C  4E75               RTS
00A29E
00A29E
00A29E
00A29E
00A29E                     SINCOS:
00A29E  3003               MOVE D3,D0
00A2A0  4EB9 00001030      JSR @COS
00A2A6  4440               NEG D0
00A2A8  0640 0100          ADD #256,D0
00A2AC  3400               MOVE D0,D2
00A2AE  4E75               RTS
00A2B0
008D78  Ende-Symboltabelle
```

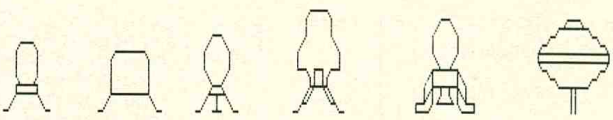


Bild 13. So sehen die Modelle aus, die in Bild 12 programmiert wurden

5. Erweitern Sie das Programm für eine Steuerung durch ein Tastenfeld, so daß man Packman nach links oder rechts gehen lassen kann.

6. Wie kann man verhindern, daß Packman bei einer Bewegung nach rechts aus dem Bildschirm verschwindet?

7. Konstruieren Sie daraus ein kleines Spiel, mit Hindernissen, die man umgehen muß.

Bild 14. Packman!

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

009C00
009C00                     ****************************
009C00                     * BEWEGTE GRAPHIK MIT DEM    *
009C00                     * FIGUR-BEFEHL  841128  RDK *
009C00                     ****************************
009C00
009C00                     ORG $9C00
009C00
009C00                     * VEREINBARUNGSBAUSTEIN
009C00
009C00                     FIG1:
009C00  00 00 00 00 01      DC.B 0,0,0,0,1,2,2,4,4,0,0,2,2
009C05  02 02 04 04 00
009C0A  00 02 02
009C0D  03 04 04 08 06      DC.B 3,4,4,8,6,9,6,4,2,0,8,2,9
009C12  09 06 04 02 00
009C17  08 02 09
009C1A  04 04 05 06 06      DC.B 4,4,5,6,6,6,6,7,10
009C1F  06 06 07 0A
009C23                     FIG2:
009C23  00 00 00 00 01      DC.B 0,0,0,0,1,2,4,4,2,2,0,0,2
009C28  02 04 04 02 02
009C2D  00 00 02
009C30  03 04 04 08 06      DC.B 3,4,4,8,6,9,6,4,2,0,8,2,9
009C35  09 06 04 02 00
009C3A  08 02 09
009C3D  04 04 05 06 06      DC.B 4,4,5,6,6,6,6,7,10
009C42  06 06 07 0A
009C46
009C46                      DS 0
009C46
009C46                     * PROGRAMMBAUSTEIN
009C46
009C46                     START:
009C46  343C 0080           MOVE #128,D2    * Y-ACHSE
009C4A  4241                CLR D1          * START BEI X=0
009C4C                     SCHLEIFE:
009C4C  41F9 00009C00       LEA FIG1,A0     * ADRESSE DER ERSTEN FIGUR
009C52  3001                MOVE D1,D0      * X PRUEFEN
009C54  0240 0008           AND #8,D0       * ALLE 8 PUNKTE WECHSELN

009C58  6700 0008           BEQ SCHL1       * =0, DANN WEITER
009C5C  41F9 00009C23       LEA FIG2,A0     * SONST ZWEITE FIGUR NEHMEN
009C62                     SCHL1:
009C62  303C 0004           MOVE #4,D0      * GROESSE DER FIGUR
009C66  4EB9 0000326C       JSR $FIGUR      * UND FIGUR AUSGEBEN
009C6C                     WARTE:           * DANN VERZOEGERN
009C6C  4EB9 00000E38       JSR $SYNC       * SONST ZU SCHNELL
009C72  6700 FFF8           BEQ WARTE
009C76  0641 0001           ADD #1,D1       * X-KOORDINATE ERHOEHEN
009C7A  0C41 0200           CMP #512,D1     * X>512, DANN STOP
009C7E  6F00 FFCC           BLE SCHLEIFE    * X<=512, DANN WIEDERHOLEN
009C82  4E75                RTS             * PROGRAMMENDE
009C84
009C84
009C84

008DD2  Ende-Symboltabelle
```

Bild 15. Packman macht den Mund auf und zu

Rolf-Dieter Klein

Tick-Tack

Das Programm „Mondlandung“ soll ergänzt werden. Es soll die genaue Zeit eines jeden Spielversuchs bestimmt werden. Aufgabe dieser Folge ist es also, eine Zeitausgabe im Bildschirm einzublenden und automatisch die Zeit zwischen Start und Stopp des Spiels anzuzeigen.

Dazu lösen wir zunächst eine Teilaufgabe, nämlich, wie man eine Uhr per Software konstruiert. Dazu kann man sich in erstem Schritt überlegen, wie denn normalerweise eine Zeitnahme erfolgt und wie zum Beispiel eine Uhr funktioniert. Da gibt es zunächst einen Taktgeber. Das kann ein Pendel sein oder eine Unruhe oder, bei modernen Uhren, ein Quarzgeber. In unserem Computer befinden sich mehrere Quarz-Taktgeber. Einmal auf der CPU-Baugruppe und dann auf der GDP-Karte. Man kann eine Zeitmessung dadurch erreichen, daß man deren Taktimpulse abzählt. Am einfachsten läßt sich der Taktgeber auf der GDP-Baugruppe dazu verwenden. Zunächst einmal schwingt der Quarz dort mit einer Frequenz von 14 MHz. Diese Frequenz ist so hoch, daß man sie auf keinen Fall per Software zählen kann. Auf der GDP-Baugruppe gibt es aber viele Taktausgänge, die eine tiefere Frequenz, abgeleitet von den 14 MHz, liefern. Am einfachsten ist der sogenannte VSYNC-Puls zugänglich, der zum Beispiel an der 7poligen Steckerleiste herausgeführt ist und einen Takt von 20 ms Periodendauer liefert. Er wird zum Synchronisieren des Bildes verwendet. Dieser Takt wird wie gesagt vom Grafik-Prozessor selbst aus dem Grundtakt abgeleitet und ist daher quarzgenau. Man kann nun den Takt beispielsweise über eine IOE-Baugruppe an einem ihrer Ports abfragen, oder, einfacher, über ein Port im Grafikprozessor selbst, dem sogenannten Statusregister auf Adresse $ffffff70. Noch einfacher ist die Verwendung eines Unterprogrammes aus dem Grundprogramm, das die Abfrage durchführt und noch ein paar zusätzliche Dinge berücksichtigt. Diese Routine heißt SYNC. Wenn man sie aufruft, so beeinflußt sie das Null-Flag (Z-Flag). Es wird gelöscht, wenn der Synchron-Puls aufgetreten ist. Wenn man die Routine ein zweites Mal aufruft, während der Puls am Port noch anliegen sollte, wird das Z-Flag gesetzt. Damit wird eine Fehlinterpretation verhindert. Die Routine SYNC garantiert also, daß das Z-Flag genau im Abstand von 20 ms gelöscht wird. Damit läßt sich eine Zeitnahme verwirklichen.

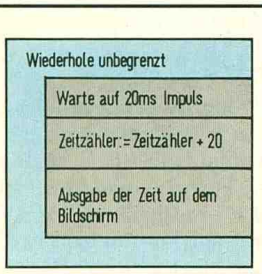


Bild 1. Das Struktogramm für die Zeitausgabe

Immer wieder: Struktogramme

Bild 1 zeigt das Struktogramm. Die Zeit soll ständig ausgegeben werden, daher wird eine Schleife, die unbegrenzt abläuft, verwendet. Dann wird auf den 20-ms-Puls gewartet. Danach kann man einen Zeitzähler erhöhen, z. B. um 20 ms, um eine Zähleinheit in Millisekunden zu erhalten.
Und dann wird die Zeit auf dem Bildschirm ausgegeben. *Bild 2* zeigt die Programmrealisierung. Als Zählregister wird D4.L verwendet. Es wird mit dem CLR-Befehl (Clear oder Löschen) auf 0 gesetzt. Dann wird die Warteschleife absolviert. Erst wenn SYNC einen Wert ungleich 0 liefert, ist der 20-ms-Puls am VSYNC-Ausgang angekommen. Danach wird der Wert 20 zum Register D4.L addiert. Nun folgt die Ausgabe des Wertes. Dazu wird zunächst die Adresse „BUFFER“ in das Adreßregister A0 geladen. Dorthin soll der Ausgabetext geschrieben werden. Der Wert D4.L wird in das Register D0.L geholt. Nun wird mit DIVU (divide unsigned oder dividiere ohne Vorzeichen) der Wert durch 100 geteilt, um eine Ausgabe in 1/10-Sekun-

```
**********************
* Uhr für Zeitnahme *
**********************

zeit:
 clr d4.l          * Zeitzähler löschen
schleife:
 jsr @sync         * warten bis 20ms
 beq schleife
 add.l #20,d4      * in d4.l sind ms
 lea buffer,a0     * Ziel für Textausgabe
 move.l d4,d0
 divu #100,d0      * aber nur  tel
 jsr @print4d      * Sek ausgeben
 lea buffer,a0     * für Ausgabe vorbereiten
 move #$22,d0      * Schriftgröße 2
 move #0,d1        * X=0
 move #220,d2      * Y=220
 jsr @write        * und Text ausgeben
 bra schleife      * wieder von vorne

buffer: ds.b 80 * Buffer für Text

Textstart=009000 Fenster=009000 Tor=009004     amer CTRL-J=Hilfe
```

Bild 2. Die Zeitnahme

den-Werten zu erreichen. Die Umrechnung in einen ASCII-Text erfolgt mit dem Unterprogramm PRINT4D, das einen Wert im Register D0.W als 16-Bit-Zahl interpretiert und als dezimale Ziffernfolge in ASCII, beginnend bei der Adresse in A0 aufsteigend ablegt. Das Ende wird durch Ablage des Wertes 0 gekennzeichnet, so wie es der WRITE-Befehl anschließend braucht. Achtung! Wenn man auf diese Weise beliebige Zahlenreihen ausgibt, so muß man den Puffer ggf. noch mit Leerzeichen auffüllen, sonst werden bei Ausgabe eines Wertes mit geringer Stellenzahl (z. B. 1) nach einer vielstelligen Zahl (z. B. 222) nur die Stellen gelöscht, die neu geschrieben werden und es erscheint eine falsche Zahl (122) auf dem Bildschirm. Bei uns ist das kein Problem, denn die Zahl wird ja immer größer (bis zu 65535, dann allerdings gibts Probleme). Die Ausgabe des Pufferinhaltes erfolgt mit WRITE. Danach zurück zur Schleife. Nach Start des Programms kann man die Zeitangabe in Zehntelsekunden verfolgen.

Unterbrechen mit Interrupt-Signalen

Um die beiden Programme Mondlandung und Zeitausgabe miteinander zu verbinden, gibt es prinzipiell zwei Möglichkeiten. Einmal kann man die Abfrage SYNC im Mondlandeprogramm verwenden und dort gleich die Zeitzählung durchführen. Die Ausgabe macht man dann am Ende jedes Durchlaufs eines Spielblocks. Die Sache hat aber möglicherweise einen Haken. Wenn man sehr komplizierte Mondlandefähren verwendet, oder das Programm um Abfragen erweitert, kann der Durchlauf des Spielblocks vielleicht länger als 20 ms dauern oder er ist sehr unregelmäßig. In beiden Fällen wäre die gemessene Zeit ungenau. Wenn man aber Interrupts verwendet, kann das nicht passieren. Interrupt bedeutet wörtlich „Unterbrechung". Man kann mit einem Interrupt den Lauf eines Programms unterbrechen. Dazu besitzt die CPU 68008 zwei Interrupteingänge, das sind Leitungen, die mit $\overline{\text{IPL0/2}}$ und mit $\overline{\text{IPL1}}$ beschriftet sind. Wenn man diese Leitungen zum Beispiel zusammen auf 0V legt, so wird ein Interrupt ausgelöst. Das laufende Programm wird unterbrochen und der Prozessor springt auf eine spezielle Adresse. Dort muß dann das sogenannte Interruptprogramm stehen, das dann abgearbeitet wird. Am Schluß des Interrupt-Programms muß ein Rücksprungbefehl folgen (ähnlich wie bei einem Unterprogramm), der Befehl RTE oder Return From Exception, Rückkehr von einer Ausnahmebehandlung. Nach dieser Rückkehr wird das unterbrochene Programm wieder weiter bearbeitet. Das Interruptprogramm sollte in unserem Fall den Zeitzähler nach jedem Interrupt um einen Wert erhöhen. Der Interrupt selbst kann zum Beispiel direkt durch die Leitung VSYNC ausgelöst werden. Im Hauptprogramm kann man dann den Zeitzähler nach jedem Spielblock-Durchlauf ausgeben, wobei man jetzt unabhängig von der Dauer eines Spieldurchlaufs ist. Interruptprogramme sind also praktisch Unterprogramme.

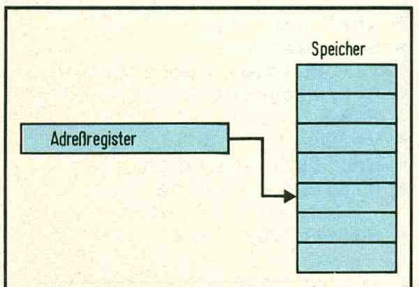


Bild 3. Der Stack wird hier von A7 verwaltet...

Was bei Interrupts zu beachten ist

Ein Interrupt-Signal kann an einer beliebigen Stelle des normalen Programmablaufs auftreten. Das Programm wird dann sofort unterbrochen. Da aber alle Register für das laufende Programm einen bestimmten Sinn haben, kann es sein, daß man wichtige Daten des Hauptprogrammes zerstört, wenn man im Interruptprogramm die Register mit neuen Werten füllt. Wenn das Interruptprogramm beendet ist, würden dann dem Hauptprogramm diese neuen, aber falschen Werte geliefert, es kommt dort zu

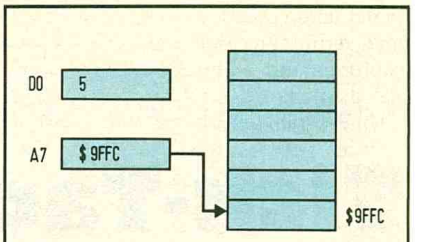


Bild 4. ...und hat den Stand $9FFC

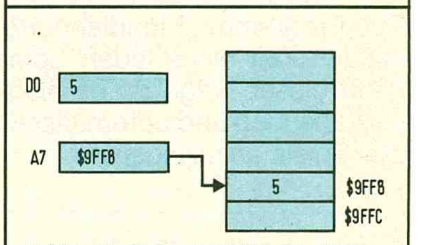


Bild 5. Nach dem MOVE.L D0,-(A7)-Befehl hat er diesen Stand

```
start:              * Programm Stackdemo
 move #5-1,d0       * hier D0 als Zähler
loop:
 move.l d0,-(a7) * Register retten
 move #50,d0        * neu besetzen
 jsr @schreite      * für Schreite
 move #72,d0        * und Drehe
 jsr @drehe
 move.l (a7)+,d0 * dann wieder zurück
 dbra d0,loop       * für die Schleife
 rts                * Programmende
```

Textstart=009000 Fenster=009000 Tor=009000 deut CTRL-J=Hilfe

Bild 6. Ein Programmbeispiel zum Stackbetrieb

fehlerhaften Ergebnissen. Das könnte man verhindern, indem man im Interruptprogramm einfach Register verwendet, die nicht im Hauptprogramm (in allen Programmteilen des Hauptprogramms) verwendet werden. Nur ist das bei uns nicht möglich, denn auch Unterprogramme, die im Hauptprogramm aufgerufen werden, verwenden Register.

Und da auch Programmteile aus dem Grundprogramm aufgerufen werden, von denen man nicht immer weiß, welche Register sie benutzen, scheidet diese Möglichkeit aus.

Ein Stack wird benötigt

Man muß also einen Weg finden, die Registerinhalte des unterbrochenen Programmes zu retten. Dazu könnte man sie einfach in Variablenplätzen abspeichern und vor Verlassen des Interruptprogramms wieder zurückladen. Besser ist aber die Verwendung eines sogenannten Stacks. Stack bedeutet Stapelspeicher. Wie bei jedem Stapel werden Elemente, die man zuletzt darauf abgelegt hat, zuerst wieder geholt. Dazu benötigt man einen Stackpointer. Wir verwenden das Register A7, ein Adreßregister, das dazu die Adresse des aktuellen Stapels enthält. Das Adreßregister muß schon mit einem sinnvollen Wert vorbelegt sein, da es auch vom Grundprogramm benötigt wird. *Bild 3* zeigt das Schema. Man kann jedes der Adreßregister des 68008 als Stackpointer verwenden, jedoch ist A7 dafür besonders geeignet. *Bild 4* zeigt nun eine Belegung. Das Register A7 habe die Adresse $9FFC gespeichert. Damit zeigt es auf eine Speicherzelle. Mit dem Befehl MOVE.L (A7),D0 kann man z. B. ein Langwort von dieser Adresse nach D0 laden. Wir nehmen aber einen anderen Befehl: MOVE.L D0,–(A7). Im Register D0 stehe z. B. der Wert 5. Nach Ausführung des MOVE-Befehls ergibt sich *Bild 5*. A7 enthält dann den Wert $9FF8 und dort steht auch der Wert 5. Das Minus vor der Klammer bei dem MOVE-Befehl bedeutet, daß vor Ausführung des Transports zunächst das Adreßregister A7 verringert wird, und zwar um den Wert der Wortbreite, also bei einem Langworttransport um den Wert 4. Danach wird der Inhalt des Registers A7 als Adresse aufgefaßt und dorthin der Inhalt des Registers D4 geschafft. Der Wert 4 steht nun auf dem Stack. Jetzt kann man D4 verändern und damit arbeitet. Am Schluß des Interrupt-Reaktionsprogrammes gibt man den Befehl: MOVE.L (A7)+,D04. Damit wird zunächst die Adresse in A7 als Quelladresse genommen und der Datenwert in dieser Zelle nach D0 transportiert. Dann wird A7 wieder um 4 erhöht. Dieser Stack, der von A7 verwaltet wird, wird übrigens auch verwendet, um die Rückkehradressen bei einem Unterprogrammaufruf zu merken. Dazu wird die Rückkehradresse bei einem BSR- oder JSR-Befehl einfach automatisch auf den Stack gelegt und das Register A7 dabei um 4 verringert. Das geschieht nur mit dem Register A7 automatisch. Daher sollte man A7 auch als Stackpointer verwenden, wenn man mit einem Stack auskommt. Bei einem RTS-Befehl wird die Rückkehradresse automatisch vom Stack geholt und der Programmzähler mit diesem Langwort geladen.

Register retten

Bild 6 zeigt ein kurzes Programmstück für die Anwendung des Rette-Befehls, der hier 5–1 aus D0 rettet. Wenn man mehrere Register retten muß, so gibt es den MOVEM-Befehl, bei dem man mehrere Register angeben kann.

Beispiel:

MOVEM.L D0–D4/D6/A2–A4,–(A7)

Die Register D0 bis D4 und D6 sowie A2 bis A4 werden auf den Stack gerettet. Am Schluß des Programmteils schreibt man

MOVEM.L (A7)+,D0–D4/D6/A2–A4,

dann werden die Register wieder vom Stack zurückgeladen. Man muß dabei darauf achten, daß man dann genau die gleiche Anzahl von Registern angibt, wie beim Eintritt ins Unterprogramm, denn sonst bleibt ein Wert auf dem Stack zurück, oder es wird einer zuviel geholt und beim nächsten RTS-Befehl geschieht Chaos, denn auf dem Stack sind auch die Rückkehradressen gespeichert. Man kann auch einzelne Worte auf dem Stack speichern (MOVE.W oder MOVEM.W).

Tabelle 1. Ausnahmevektoren beim 68000/68008

Vektor-nummer	Adresse	Bedeutung
0	0000	Reset: Stackpointer
1	0004	Reset: Programmzähler
2	0008	Bus Error (wenn BERR)
3	000C	Address Error (ungerade Adresse)
4	0010	Illegal Instruction (Befehl unbekannt)
5	0014	Zero Divide (Division durch 0)
6	0018	CHK-Instruktion (Bereichsgrenzen)
7	001C	TRAPV-Inst. (Überlauf bei TRAPV)
8	0020	Privilege Violation (privilegierter Befehl)
9	0024	Trace (Trace ist aktiv, nach Befehl)
10	0028	LINE 1010 Emulator
11	002C	LINE 1111 Emulator
12	0030	für Erweiterungen reserviert
13	0034	für Erweiterungen reserviert
14	0038	für Erweiterungen reserviert
15	003C	Uninitialized Interrupt Vektor
16–23	0040	für Erweiterungen reserviert
	005F	–
24	0060	Spurious Interrupt (bei BERR + INT)
25	0064	Level 1 Interrupt Autovector (68000)
26	0068	Level 2 Interrupt Autovector (68008) NMI
27	006C	Level 3 Interrupt Autovector (68000)
28	0070	Level 4 Interrupt Autovector (68000)
29	0074	Level 5 Interrupt Autovector (68008) INT
30	0078	Level 6 Interrupt Autovector (68000)
31	007C	Level 7 Interrupt Autovector (68008) NMI+INT
32–47	0080	Trap-Instruction Vectors
	00BF	–
48–63	00C0	für Erweiterungen reserviert
	00FF	–
64–256	0100	User-Interrupt-Vektoren
	03FF	–

Im Interruptprogramm speichere man also zuerst alle Registerinhalte von den Registern ab, die man darin verwenden will, und lade sie vor dem RTE-Befehl wieder zurück. Es gibt noch das Statusregister, in dem alle Flags gespeichert sind. Dieses Register wird aber vom Prozessor bei einem Interrupt automatisch auf den Stack gerettet und auch beim Rücksprung muß man sich nicht weiter darum kümmern.

Ausnahmen

Ein Interrupt gehört zu den sogenannten Exceptions, also den Ausnahmebedingungen. Es gibt beim 68008 eine Vielzahl davon. Zum Beispiel der schon oft erwähnte ADDRESS ERROR, der immer dann auftritt, wenn der Prozessor einen Wort- oder Langwortzugriff auf eine ungerade Adresse durchführt. Diese Exceptions werden vom Grundprogramm automatisch behandelt. Es erscheint eine passende Meldung auf dem Bildschirm. *Tabelle 1* zeigt eine Zusammenstellung der zugehörigen Vektoren.

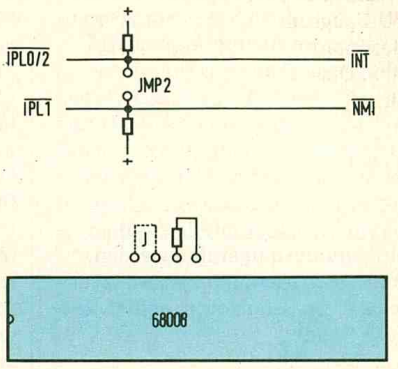


Bild 8.
Die Brücke JMP 2

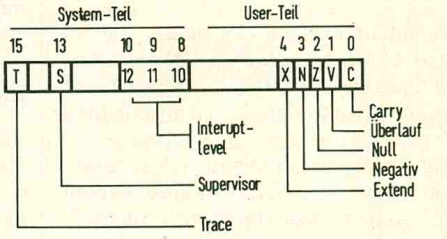


Bild 9.
Das Statusregister des 68008

```
                        151:
                        152:
                        153: * Speichergebiet fuer Variable etc.
                        154:
008000                  155:          offset $8000    * start RAM-Gebiet
                        156:
                        157:
                        158: ****** byte,wort,lang-variable *** User-Verwendbar
00008000                159: vekdest:                  * INIT-STart  fuer transport
                        160:                           * dort Sprueunge eintragen JMP xxxx.l
00008000                161: intlv2:        ds.b 6     * Auto-Interrupt LV2 (ltg nmi)
00008006                162: intlv5:        ds.b 6     * Auto-Interrupt LV5 (ltg int)
0000800C                163: intlv7:        ds.b 6     * Auto-Interrupt LV7 (ltg nmi+int)
00008012                164: trap0:         ds.b 6     * TRAP0-Vektor
                        165:
00008018                166: userci:        ds.b 6     * Vektor USER CI
0000801E                167: usercsts:      ds.b 6     * Vektor USER CSTS
00008024                168: userco:        ds.b 6     * Vektor USER CO
                        169:
                        170: **** gebiet oberhalb nicht veraendern ohne init: zu beachten
                        171:
0000802A                172: iostat:        ds.b 1     * =1 dann co2 Ausgabe auf CRT
0000802B                173: iostatb:       ds.b 1     * fuer ci2,Umschalter
0000802C                174: iodir:         ds.b 1     * fuer CTRL-A Umschaltung
00802D                  175:                ds.b 1     * Angleich
                        176:
0000802E                177: flip:          ds.b 1     * fuer autoflip
0000802F                178: flipcnt:       ds.b 1
00008030                179: flip1:         ds.b 1
00008031                180: flip1cnt:      ds.b 1
                        181:
                        182:
00008032                183: passflag:      ds.b 2     * Aktueller PASS beim Assembler
00008034                184: errflag:       ds.b 2     * Fehlerflag 0=kein Fehler bei Ass
                        185:
                        186: * editor
                        187:
00008036                188: stxtxt:        ds.b 4     * Zeiger auf Textanfang
0000803A                189: akttxt:        ds.b 4     * Zeiger auf Startzeile des Bildschirms
0000803E                190: etxtxt:        ds.b 4     * Zeiger auf Textende
00008042                191: eottxt:        ds.b 4     * Zeiger auf das Ende aller Texte
                        192:
                        193:
                        194: *** lokale Variable und Nicht-Benutzer-Zellen
                        195:
                        196: ****** wort,lang-variable
                        197:
                        198:
00008046                199: turx:          ds.b 2     *x-pos turtle *16
00008048                200: tury:          ds.b 2     *y-pos turtle *32
0000804A                201: turphi:        ds.b 2     *winkel
0000804C                202: turlx:         ds.b 2     *merker
0000804E                203: turly:         ds.b 2
```

Bild 7. Die RAM-Belegung des Grundprogramms

Es gibt nun viele Interruptmöglichkeiten beim 68008. Wir verwenden hier nur drei besonders einfache. Die Interrupts Autolevel 2,5 und 7. Wenn zum Beispiel die Leitung NMI am 68008 auf 0V geht und der Interrupt freigegeben ist, so holt sich der Prozessor von Adresse $68 ein Langwort. Dieses Langwort interpretiert er als Adresse des Interruptprogrammanfangs. Bei der Leitung INT wird der Vektor von Adresse $74 geholt und, wenn man beide Leitungen gleichzeitig auf 0V legt, von Adresse $7C.

Auf diesen Adressen stehen beim Grundprogramm Werte mit Adressen, die ins RAM führen. Da können Sie eigene Programme ablegen, um individuell zu reagieren. *Bild 7* zeigt einen Ausschnitt aus der RAM-Belegung des Grundprogramms. Dort sind jeweils 6 Bytes für die Befehle reserviert, die nach einem Interrupt dieser Art angesprungen werden. Bei der Marke INTLV7 kann man zum Beispiel direkt einen Sprung dorthin schreiben, denn ein absoluter Sprung belegt genau 6 Bytes. Die anderen Interruptarten des 68008 sind nicht verwendet.

Wenn man auch noch $\overline{IPL1}$ damit verbindet, also z. B. die Brücke JMP2 auf der CPU-Baugruppe einlötet (*Bild 8*), so wird der Interrupt-Autovektor 7 angesprungen. Dieser Vektor hat eine Besonderheit.

Maskieren

Man unterscheidet zwei Arten von Interrupts, nämlich die maskierbaren und die nicht-maskierbaren Interrupts. Der Autovektor 7 ist nichtmaskierbar, das heißt, ein solcher Interrupt wird immer ausgelöst. Ferner wird er nur dann ausgelöst, wenn ein Wechsel von 1 auf 0 an den Leitungen stattfindet. Er reagiert also nur auf eine Signal-Flanke, während die anderen auf statische Signale reagieren (wenn die Leitungen auf 0 V liegen). Die maskierbaren Interrupts reagieren auch nur, wenn man im Statusregister die Interrupts freigibt, indem man die sogenannte Unterbrechungsebene verändert, die normalerweise auf 7 geschaltet ist und dann Interrupts mit niedrigeren Nummern nicht zuläßt. *Bild 9* zeigt die Belegung des Statusregisters, das auch das Flagregister enthält. Die Bits 15 bis 8 sind die sogenannten Systembits. Bit 7 bis 0 sind die User-Bits.

Die Betriebsarten des 68008

Im 68008 unterscheidet man zwei Betriebsarten, den Systembetrieb und den Userbetrieb. Bit 13 ist auf 1, wenn der Systembetrieb eingeschaltet ist. Dann können alle Befehle des 68008 ausgeführt werden. Im sogenannten User-Modus geht das nicht, es gibt dort privilegierte Befehle, die zu einer Exception führen, wenn ein Programm im User-Modus sie enthält. (Im Grundprogramm werden auch die User-Programme im System-Mode ausgeführt). Man kann den User-Modus einschalten, wenn man das Bit 13 im Statusregister auf 0 setzt. Den Interruptlevel kann man ebenfalls ändern.

Eine Veränderung der genannten Bits ist mit MOVE, EOR, AND und OR-Befehlen möglich, wenn man als Ziel SR angibt.

MOVE #$2700,SR setzt zum Beispiel den Interruptlevel auf 7 zurück. Achtung, wenn man das Statusregister im Einzelschritt betrachtet, so ist auch Bit 15 gesetzt, das Trace-Bit, das es gestattet, Befehle im Einzelschritt zu durchlaufen: Wenn man das Tracebit setzt, so wird ein Befehl ausgeführt und anschließend eine Exception auf den Trace-Vektor durchgeführt.

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1
009C00                  ;*********************************
009C00                  ; MONDLANDEPROGRAMM VERSION 1.0 ;
009C00                  ; VOM 13.12.1983                ;
009C00                  ;*********************************
009C00
009C00                  ; ***** INTERRUPT-BAUSTEIN *****
009C00
00800C                   ORG $800C ; FESTE ADRESSE
00800C
00800C  4EF9 0000A0A2    JMP INTAUS ; DORTHIN SPRUNG
008012
00A000                   ORG $A000
00A000
00A000                  ; ***** VEREINBARUNGS-BAUSTEIN *****
00A000
= FFFFFF30              TASTEN EQU $FFFFFF30
= 00000001              RECHTS EQU 1   ; BIT MASKEN
= 00000002              LINKS  EQU 2
= 00000004              RAUF   EQU 4
= 00000008              RUNTER EQU 8
00A000
00A000                  ERFOLGTEXT:
00A000  4775742067656C   DC.B 'Gut gelandet',0
00A007  616E64657400
00A00D
00A00D                  BRUCHTEXT:
00A00D  42727563686C61   DC.B 'Bruchlandung',0
00A014  6E6475AFA700
00A01A
00A01A                  FAEHRE:
00A01A  08 02 09 00 00   DC.B 8,2,9,0,0,7,3,2,2,4,4,4,4
00A01F  07 03 02 02 04
00A024  04 04 04
00A027  06 06 05 01 00   DC.B 6,6,5,1,0,0,10
00A02C  00 0A
00A02E
00A02E                   DS 0 ; ANGLEICH AUF WORTGRENZE
00A02E
00A02E                  ; VARIABLENVEREINBARUNGEN
00A02E
00A02E  0000            X: DC.W  0
00A030  0000            Y: DC.W  0
00A032  0000            DX: DC.W  0
00A034  0000            DY: DC.W  0
00A036  0000            TREIBST: DC.W  0
00A038
00A038  00000000        TIMERMS: DC.L 0
00A03C  0000            TIMER10TEL: DC.W  0
00A03E                  BUFFER: DS 50
00A0A2
00A0A2                  ; ***** UNTERPROGRAMM-BAUSTEIN *****
00A0A2
00A0A2                  INTAUS:
00A0A2  48E7 1000        MOVEM.L D3,-(A7)
00A0A6  2639 0000A038    MOVE.L TIMERMS,D3
00A0AC  0683 00000014    ADD.L #20,D3
00A0B2  23C3 0000A038    MOVE.L D3,TIMERMS
00A0B8  86FC 0064        DIVU #100,D3 ;  UMRECHNEN IN 10TEL SEC
00A0BC  33C3 0000A03C    MOVE.W  D3,TIMER10TEL
00A0C2  4CDF 0008        MOVEM.L (A7)+,D3
00A0C6  4E73             RTE
00A0C8
00A0C8
00A0C8                  ZEITAUS:
00A0C8  41F9 0000A03E    LEA BUFFER,A0
00A0CE  3039 0000A03C    MOVE.W  TIMER10TEL,D0
00A0D4  4EB9 000015EA    JSR @PRINT4D
00A0DA  41F9 0000A03E    LEA BUFFER,A0
00A0E0  303C 0022        MOVE #$22,D0
00A0E4  323C 0000        MOVE #0,D1
00A0E8  343C 00DC        MOVE #220,D2
00A0EC  4EB9 00001386    JSR @WRITE
00A0F2  4E75             RTS
00A0F4
00A0F4
00A0F4
00A0F4                  MONDFLAECHE:
00A0F4  323C 0000        MOVE #0,D1
00A0F8  343C 0000        MOVE #0,D2
00A0FC  363C 005A        MOVE #90,D3
00A100  4EB9 0000131A    JSR @SET       ;SETZE SCHILDKROETE
00A106  4EB9 00001112    JSR @HIDE      ;ABER NICHT ZEIGEN
00A10C  383C 0031        MOVE #50-1,D4     ;ANZAHL KRATER
```

Bild 10. Das Mondlandeprogramm mit Zeitnahme

```
00A110                  MLP1:
00A110  4EB9 0000A11C    JSR KRATER
00A116  51CC FFF8        DBRA D4,MLP1
00A11A  4E75             RTS
00A11C
00A11C                  KRATER:
00A11C  303C FFD3        MOVE  #-45,D0
00A120  4EB9 000012F6    JSR @DREHE
00A126  303C 000A        MOVE #10,D0
00A12A  4EB9 00001288    JSR @SCHREITE
00A130  303C FFD3        MOVE #-45,D0
00A134  4EB9 000012F6    JSR @DREHE
00A13A  303C 0006        MOVE #6,D0
00A13E  4EB9 00001288    JSR @SCHREITE
00A144  303C FFD3        MOVE #-45,D0
00A148  4EB9 000012F6    JSR @DREHE
00A14E  303C 000A        MOVE #10,D0
00A152  4EB9 00001288    JSR @SCHREITE
00A158  303C FF1F        MOVE #-45-180,D0
00A15C  4EB9 000012F6    JSR @DREHE
00A162  4E75             RTS
00A164
00A164                  ; ***** PROGRAMM-BAUSTEIN *****
00A164
00A164                  START:
00A164  6100 FF8E        BSR MONDFLAECHE
00A168  33FC 0064        MOVE #100,X
00A16C  0000A02E
00A170  33FC 00C8        MOVE #200,Y
00A174  0000A030
00A178  33FC 0003        MOVE #3,DX
00A17C  0000A032
00A180  33FC 0000        MOVE #0,DY
00A184  0000A034
00A188  33FC 012C        MOVE #300,TREIBST
00A18C  0000A036
00A190  42B9 0000A038    CLR.L TIMERMS          ; INITIALISIEREN
00A196  4279 0000A03C    CLR TIMER10TEL
00A19C                  WDH1:
00A19C  6100 FF2A        BSR ZEITAUS
00A1A0  207C 0000A01A    MOVEA.L #FAEHRE,A0
00A1A6  303C 0003        MOVE #$3,D0            ; VERGROESSERUNG=3
00A1AA  3239 0000A02E    MOVE X,D1
00A1B0  3439 0000A030    MOVE Y,D2
00A1B6  4EB9 0000326C    JSR @FIGUR
00A1BC  4A79 0000A036    TST TREIBST
00A1C2  6F00 0052        BLE NOTEST             ; KEIN TREIBSTOFF ?
00A1C6  1039 FFFFFF30    MOVE.B TASTEN,D0
00A1CC  0200 0001        AND.B #RECHTS,D0
00A1D0  6608             BNE.S TST1
00A1D2  5679 0000A032    ADDQ #3,DX
00A1D8  603C             BRA.S NOTEST
00A1DA
00A1DA                  TST1:
00A1DA  1039 FFFFFF30    MOVE.B TASTEN,D0
00A1E0  0200 0002        AND.B #LINKS,D0
00A1E4  6600 000A        BNE TST2
00A1E8  5779 0000A032    SUBQ #3,DX
00A1EE  6026             BRA.S NOTEST
00A1F0
00A1F0                  TST2:
00A1F0  1039 FFFFFF30    MOVE.B TASTEN,D0
00A1F6  0200 0004        AND.B #RAUF,D0
00A1FA  6608             BNE.S TST3
00A1FC  5679 0000A034    ADDQ #3,DY
00A202  6012             BRA.S NOTEST
00A204
00A204                  TST3:
00A204  1039 FFFFFF30    MOVE.B TASTEN,D0
00A20A  0200 0008        AND.B #RUNTER,D0
00A20E  6606             BNE.S NOTEST
00A210  5779 0000A034    SUBQ #3,DY
00A216
00A216                  NOTEST:
00A216  5379 0000A034    SUBQ #1,DY             ;GRAVITATION
00A21C  3639 0000A032    MOVE DX,D3             ; X:=X+DX
00A222  D779 0000A02E    ADD D3,X
00A228  3639 0000A034    MOVE DY,D3             ; Y:=Y+DY
00A22E  D779 0000A030    ADD D3,Y
00A234  5379 0000A036    SUBQ #1,TREIBST        ; TREIBSTOFF VERINGERN
00A23A
00A23A  323C 0004        MOVE #5-1,D1           ; WARTESCHLEIFE
00A23E                  WARTE20MS:
00A23E  3E3C 001C        MOVE #!SYNC,D7
00A242  4E41             TRAP #1
00A244  6700 FFF8        BEQ WARTE20MS
00A248  51C9 FFF4        DBRA D1,WARTE20MS
00A24C
00A24C  0C79 0005        CMP #5,Y               ; HOEHE DER OBERFLAECHE UEBER 0
00A250  0000A030
00A254  6E00 FF46        BGT WDH1               ; BERUECKSICHTIGEN
00A258                                          ;
00A258  303C 0003        MOVE #$3,D0            ; FAEHRE AM SCHLUSS NOCHMALS ZEIGEN
00A25C  3239 0000A02E    MOVE X,D1
00A262  3439 0000A030    MOVE Y,D2
00A268  4EB9 0000326C    JSR @FIGUR             ; FIGUR AUFRUFEN
00A26E  3039 0000A032    MOVE DX,D0
00A274  6A02             BPL.S FF1
00A276  4440             NEG D0
00A278                  FF1:                    ; !DX! < 2
00A278  0C40 0002        CMP #2,D0
00A27C  6C18             BGE.S BRUCH
00A27E  3039 0000A034    MOVE DY,D0
00A284  6A02             BPL.S FF2
00A286  4440             NEG D0
00A288                  FF2:                    ; !DY! < 4
00A288  0C40 0004        CMP #4,D0
00A28C  6C0B             BGE.S BRUCH
00A28E  41F9 0000A000    LEA ERFOLGTEXT,A0
00A294  6006             BRA.S ENDE
00A296
00A296                  BRUCH:
00A296  41F9 0000A00D    LEA BRUCHTEXT,A0
00A29C                  ENDE:
00A29C  303C 0033        MOVE #$33,D0           ; SCHRIFTGROESSE
00A2A0  323C 0000        MOVE #0,D1             ; X-POSITION
00A2A4  343C 0082        MOVE #130,D2           ; Y-POSITION
00A2A8  4EB9 00001386    JSR @WRITE
00A2AE  6100 FE18        BSR ZEITAUS            ; LETZTER STAND
00A2B2  4EB9 00000A00    JSR @CI                ; WARTEN AUF TASTE
00A2B8  4EB9 00000DA0    JSR @CLR               ; LOESCHEN
00A2BE  6000 FEA4        BRA START              ; UNBEGRENZT
00A2C2
00A2C2                  ; ***********
00A2C2
00A2C2                  END
008DD2  Ende-Symboltabelle
```

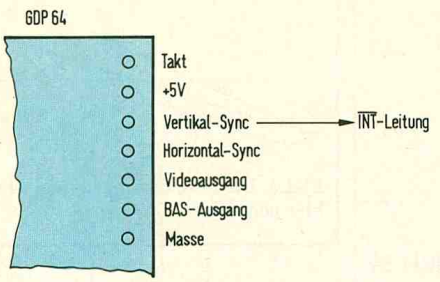


Bild 11. So wird VSYNC mit INT verbunden

Im User-Modus kann man nur noch die Bedingungsflags setzen, also z. B. mit MOVE ..., CCR, der schreibende Zugriff auf das Statusregister ist verboten. Wer ausführliche Informationen über die Exception-Behandlung braucht, sollte sich im Handbuch zum 68008/68000 einlesen, das alles ausführlich zu schildern, das würde zu weit führen.

Die Zeitmessung

Bild 10 zeigt das vollständige Programm „Mondlandung mit Zeitnahme". Als erstes beginnt es mit dem Interrupt-Baustein. Mit dem ORG $800C-Befehl wird erreicht, daß der nachfolgende Befehl, ein Sprung, an die Stelle im RAM gelegt wird, auf die der Sprung beim Interrupt Level 7 automatisch gerichtet ist. Dort kann man aber nur diesen Sprung hinlegen, das eigentliche Interruptprogramm muß an anderer Stelle stehen. Daher kommt nun eine weitere ORG-Anweisung. Dann folgt der Vereinbarungsbaustein, wie schon in der letzten Folge beschrieben. Neu ist, neben ein paar Variablen, der Unterprogrammbaustein.

Dort gibt es die Routine INTAUS, die alle 20 ms angesprungen wird. Das Register D3 wird hier gerettet, da es im Programmteil benutzt wird. Dann wird der alte Wert der Zeit in TIMERMS geladen, 20 addiert, und der neue Wert zurückgespeichert. Für die Ausgabe wird der Wert durch 100 dividiert, denn eine Ausgabe in 10tel-Sekunden genügt uns.

Dieser Wert wird in TIMER10TEL gespeichert. Danach wird der alte Inhalt von D3 aus dem Stack zurückgeholt.

Mit RTE wird das Interruptprogramm beendet. RTE bedeutet Return From Exception und unterscheidet sich vom RTS bei Unterprogrammen dadurch, daß es das alte Statusregister wieder zurückholt, dessen Inhalt von der CPU automatisch beim Interrupt gerettet wurde. Also nicht RTS verwenden, sonst stürzt das Programm ab.

Alles andere ist bekannt

Die anderen Programmteile sind bekannt. Die Zeitausgabe erfolgt im Hauptprogramm einmal gleich zu Beginn des Spielblocks mit ZEITAUS und dann am Schluß des Programms, vor dem Aufruf von CI. Hinweis zur Inbetriebnahme: Bevor man die VSYNC-Leitung an den Bus anschließt, muß das Programm übersetzt sein, denn sonst würde ein Interrupt ausgelöst, aber es wäre gar kein Interruptprogramm vorhanden, was zum Absturz führen würde. Daher am besten einen Schalter verwenden, der den Interrupt aktiviert. So können Sie vorgehen:

1. Gemäß *Bild 11* eine Verbindung vom VSYNC-Ausgang der GDP zu $\overline{\text{INT}}$ des Busses durchführen. Am besten einen Schalter dazwischenlegen und auf „aus" schalten.

2. Brücke an der CPU einsetzen: JMP 2.

3. Programm eingeben, aber die Interruptleitung darf noch nicht verbunden werden, sonst Absturz!

4. Programm übersetzen.

5. Interrupt einschalten (Verbindung VSYNC --- INT herstellen).

6. Programm starten, die Zeit muß links oben sichtbar werden.

Wenn man jetzt das Grundprogramm wieder aufruft, z. B. durch Reset, so bleibt der Interrupt aktiv. Man kann durch „ANSEHEN" oder mit „IO LESEN" die Speicherzelle „TIMERMS" ansehen und wird feststellen, daß sie ihren Wert ständig ändert.

Aufgaben

1. Wie kann man eine Uhrzeit ins Grundprogramm einblenden?
Hinweis: Legen Sie dazu auch die Ausgaberoutine ins Interruptprogramm. Achtung: Alle Register des GDPs müssen nun auch gerettet werden, denn er ist, wie die CPU, auch betroffen, wenn man Ausgaben im Interrupt-Modus macht. Es sind dies die Ports $ffffff71 bis $ffffff7f, man rettet sie z. B. durch: (MOVE.B $ffffff71.D0 -- MOVE.W D0,–(A7)). Am Schluß müssen die Register aber wieder zurückgeladen werden und vor dem Retten muß man WAIT als Unterprogramm aufrufen., Das Register am Port $FFFFF70 wird nicht gerettet, da es beim Lesen anders reagiert als beim Schreiben (Befehlsport).

2. Was passiert, wenn man den Interrupt-Autolevel 5 verwendet? (Brücke JMP 2 offenlassen und SR mit einem kleineren Level belegen ($2000 z. B.), ORG $8006 verwenden). Warum stimmt die Zeitnahme nicht mehr? Was muß man schaltungstechnisch bei der Interruptauslösung ändern, oder was im Interruptprogramm?

3. Denken Sie sich bitte weitere Einsatzmöglichkeiten für Interrupts aus.

Rolf-Dieter Klein

Ausflug nach Simland

Die Simulation dreidimensionaler Vorgänge ist ein Einsatzgebiet für Computer. So werden zum Beispiel Flugsimulatoren verwendet, um Piloten zu schulen. Man kann so gefährliche Situationen gefahrlos meistern und üben. In manchem Flugsimulator wird neben der Simulation der Anzeigen und Bewegungen auch die Simulation des Blickes durch das Cockpitfenster durchgeführt. Diese Aufgabe ist sehr anspruchsvoll, denn die Bewegungen des Flugzeugs sollen ja in Echtzeit dargestellt werden.

Bei Flugsimulatoren verwendet man hochgezüchtete Spezialrechner, die nur für diesen Zweck konstruiert wurden. Da der NDR-Klein-Computer über einen schnellen Grafikprozessor und eine leistungsfähige CPU verfügt, sollte er solche Simulationen ebenfalls andeuten können. In dieser Folge soll einmal die dreidimensionale Projektion eines Gegenstands auf den Bildschirm programmiert werden, den man im Raum bewegen kann. Das Programm läßt sich dann auch zu einem Flugsimulator ausbauen.

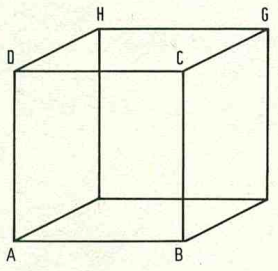


Bild 1. Der Würfel, wie er dargestellt werden soll

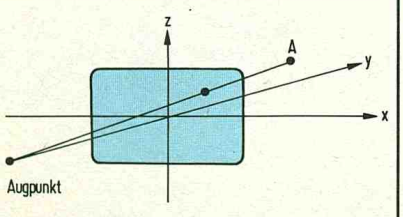


Bild 2. Zur Projektion auf den Bildschirm

Mathematik

Zuerst ein paar wichtige Begriffe. Der dreidimensionale Körper wird in Form von Koordinaten (x, y, z) in den Computer eingegeben. Wenn er auf dem Bildschirm dargestellt werden soll, wird eine Projektion durchgeführt. Hier wird die Zentralprojektion verwendet.

Bild 1 zeigt den Würfel, wie er auf dem Bildschirm erscheint. In *Bild 2* ist ein Projektionsvorgang skizziert. Wir benötigen dazu ein Koordinatenkreuz. Die x-Achse zeige nach rechts, die z-Achse nach oben und die y-Achse nach hinten (so ist es in der Mathematik gebräuchlich). In der x-z-Ebene soll der Bildschirm liegen. Der Betrachter befinde sich in einigem Abstand vor dem Bildschirm, dort wo der sogenannte Augpunkt sich befindet. Wenn man von da aus den Punkt A des Würfels beobachtet, so verläuft eine Sichtlinie vom Augpunkt geradlinig zum Punkt A. Sie schneidet die Bildebene in einem Punkt, dem Projektionspunkt. Seine Lage muß aus den Koordinaten von A berechnet werden.

Bild 3 zeigt die verwendeten Bezeichnungen. Der Betrag von y0 ist der Abstand des Augpunktes vom Nullpunkt. y0 ist im Programm mit einer negativen Zahl anzugeben.

xf und zf bezeichnen den Projektionspunkt. x, y und z sind die Koordinaten des zu projezierenden Punktes A. Dann ergibt sich folgende Formel:

$$xf = \frac{y0 * x}{y0 - y} \qquad zf = \frac{y0 * z}{y0 - y}$$

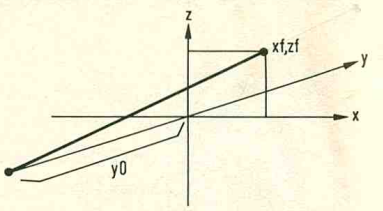


Bild 3. Die Bezeichnungen, wie wir sie hier benutzen

Eine Eigenschaft dieser Formel: Wenn der Augpunkt sehr weit vom Bildschirm entfernt wird, also der Betrag von y0 groß wird (y0 ist immer negativ), dann wird xf ≅ x und zf ≅ z (man erhält beinahe eine Parallelprojektion).

Der Programmentwurf

Wenn man einen Gegenstand auf dem Bildschirm darstellen will, so muß man die Formel oben für jeden Punkt des Gegenstandes ausrechnen. Doch gibt es eine abkürzende Form. Bei einem Würfel zum Beispiel genügt es, die Formel auf die Eckpunkte anzuwenden und dann die projezierten Eckpunkte miteinander zu verbinden. Denn Geraden gehen bei Zentralprojektion in Geraden über. Dieser Umstand spart viel Rechenaufwand.
Ein Struktogramm zeigt *Bild 4*. Pro Kante des Würfels oder anderen Gegenstands mit geraden Kanten werden zwei Koordinatenpaare definiert. Daraus werden jeweils die Projektionspunkte ermittelt und dann deren Verbindungslinie gezeichnet. So verfährt man, bis alle Kanten berechnet sind. Die Kanten kann man einfach in einer Tabelle unterbringen. *Bild 5* zeigt das Programm und *Bild 6* die Ausgabe auf dem Bildschirm.

Perspektiven

Im Programm gibt es ein paar Konstanten, die noch nicht erklärt wurden. PERSP stellt die Augenhöhe ein. Damit kann man unterschiedliche Perspekti-

Bild 4. Das Ausgabestruktogramm

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

009C00                  ;************************
009C00                  ; AUSFLUG NACH SIMLAND  ;
009C00                  ; VERSION 1.0  RDK      ;
009C00                  ; MIT PERSPEKTIVE       ;
009C00                  ;************************
009C00
00A000                  ORG $A000
00A000
00A000                  ; ***** VEREINBARUNGS-BAUSTEIN *****
00A000
00A000                  ; WUERFEL - VERBINDUNGSLINIEN
00A000
00A000                  WUERFEL:
00A000  FF9C FF9C FF9C  DC.W -100,-100,-100,100,-100,-100
00A006  0064 FF9C FF9C
00A00C  FF9C FF9C FF9C  DC.W -100,-100,-100,-100,-100,100
00A012  FF9C FF9C 0064
00A018  FF9C FF9C 0064  DC.W -100,-100,100,100,-100,100
00A01E  0064 FF9C 0064
00A024  0064 FF9C 0064  DC.W 100,-100,100,100,-100,-100
00A02A  0064 FF9C FF9C
00A030  0064 FF9C FF9C  DC.W 100,-100,-100,100,100,-100
00A036  0064 0064 FF9C
00A03C  FF9C FF9C FF9C  DC.W -100,-100,-100,-100,100,-100
00A042  FF9C 0064 FF9C
00A048  FF9C FF9C 0064  DC.W -100,-100,100,-100,100,100
00A04E  FF9C 0064 0064
00A054  0064 FF9C 0064  DC.W 100,-100,100,100,100,100
00A05A  0064 0064 0064
00A060  FF9C 0064 FF9C  DC.W -100,100,-100,100,100,-100
00A066  0064 0064 FF9C
00A06C  FF9C 0064 FF9C  DC.W -100,100,-100,-100,100,100
00A072  FF9C 0064 0064
00A078  FF9C 0064 0064  DC.W -100,100,100,100,100,100
00A07E  0064 0064 0064
00A084  0064 0064 0064  DC.W 100,100,100,100,100,-100
00A08A  0064 0064 FF9C
00A090
= 0000000C              ANZAHL EQU 12 ; ANZAHL KANTEN
= 0000000C              KOOR   EQU 12 ; ANZAHL BYTES PRO EINTRAG
00A090
= 00000000              X1 EQU 0  ; OFFSET-ADRESSIERUNG
= 00000002              Y1 EQU 2  ; WORTDISTANZ
= 00000004              Z1 EQU 4
= 00000006              X2 EQU 6
= 00000008              Y2 EQU 8
= 0000000A              Z2 EQU 10
00A090
= FFFFFC18              Y0 EQU -1000 ; AUGPUNKT
= FFFFFF01              PERSP EQU -255 ; AUGHOEHE
00A090
00A090  0000            XF1: DC.W 0  ; VARIABLE
00A092  0000            ZF1: DC.W 0
00A094  0000            XF2: DC.W 0
00A096  0000            ZF2: DC.W 0
00A098
00A098
00A098                  ; ***** UNTERPROGRAMM-BAUSTEIN *****
00A098
00A098
00A098
00A098
00A098
00A098
00A098                  ; ***** PROGRAMM-BAUSTEIN *****
00A098
00A098                  START:
00A098  41F9 0000A000    LEA WUERFEL,A0  ; ADRESSE DER TABELLE
00A09E  3A3C 000B        MOVE #ANZAHL-1,D5 ; SCHLEIFENZAEHLER
00A0A2                  WDH1:
00A0A2  3028 0000        MOVE X1(A0),D0
00A0A6  C1FC FC18        MULS #Y0,D0
00A0AA  323C FC18        MOVE #Y0,D1
00A0AE  9268 0002        SUB Y1(A0),D1
00A0B2  81C1             DIVS D1,D0 ; XF1 BERECHNET
00A0B4  33C0 0000A090    MOVE D0,XF1
00A0BA  3028 0004        MOVE Z1(A0),D0
00A0BE  0640 FF01        ADD #PERSP,D0
00A0C2  C1FC FC18        MULS #Y0,D0
00A0C6  81C1             DIVS D1,D0 ; ZF1 BERECHNET
00A0C8  33C0 0000A092    MOVE D0,ZF1
00A0CE  3028 0006        MOVE X2(A0),D0
00A0D2  C1FC FC18        MULS #Y0,D0
00A0D6  323C FC18        MOVE #Y0,D1
00A0DA  9268 0008        SUB Y2(A0),D1
00A0DE  81C1             DIVS D1,D0
00A0E0  33C0 0000A094    MOVE D0,XF2 ; XF1 BERECHNET
00A0E6  3028 000A        MOVE Z2(A0),D0
00A0EA  0640 FF01        ADD #PERSP,D0
00A0EE  C1FC FC18        MULS #Y0,D0
00A0F2  81C1             DIVS D1,D0
00A0F4  33C0 0000A096    MOVE D0,ZF2
00A0FA  3239 0000A090    MOVE XF1,D1
00A100  3439 0000A092    MOVE ZF1,D2
00A106  0641 0100        ADD #256,D1  ; IN BILDSCHIRMITTE BRINGEN
00A10A  0642 0100        ADD #256,D2
00A10E  0442 FF01        SUB #PERSP,D2
00A112  48C2             EXT.L D2
00A114  85FC 0002        DIVS #2,D2
00A118  4EB9 00000EE2    JSR @MOVETO
00A11E  3239 0000A094    MOVE XF2,D1
00A124  3439 0000A096    MOVE ZF2,D2
00A12A  0641 0100        ADD #256,D1 ; IN BILDSCHIRMMITTE BRINGEN
00A12E  0442 FF01        SUB #PERSP,D2
00A132  0642 0100        ADD #256,D2
00A136  48C2             EXT.L D2
00A138  85FC 0002        DIVS #2,D2
00A13C  4EB9 00000F4C    JSR @DRAWTO
00A142
00A142  D1FC 0000000C    ADDA.L #KOOR,A0
00A148  51CD FF58        DBRA D5,WDH1 ; BIS ALLE KANTEN BERECHNET
00A14C
00A14C  4E75            RTS
00A14E
00A14E
00A14E                  ; ********
00A14E                   END

008B80  Ende-Symboltabelle
```

Bild 5. Das Programm für die perspektivische Projektion

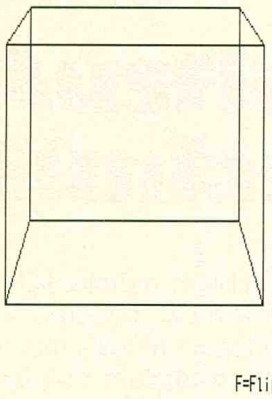


Bild 6. So sieht der Würfel auf dem Bildschirm aus

ven einstellen: Die Froschperspektive, wenn man den Augpunkt unterhalb des Gegenstandes anordnet, die Zentralperspektive, wenn man den Augpunkt mittig einstellt (PERSP=0) und die Vogelperspektive, wenn der Augpunkt höher als das Objekt liegt.
Man erreicht das Verschieben des Augpunktes einfach dadurch, daß man vor der Projektion einfach den ganzen Gegenstand nach oben oder unten verschiebt und nach der Projektion das Ergebnis wieder in die Bildmitte schafft, damit es sichtbar bleibt.
Ein Würfel wird durch seine Kanten gut definiert. Daher benötigt man 12 Einträge, denn ein Würfel besitzt 12 Kanten. Jeder Eintrag besteht aus einem Koordinatenpaar x1, y1, z1 und x2, y2 und z2.

Im Hauptprogramm wird zur Adressierung dieser Tabelle ein Adreßregister verwendet. Dadurch kann man der Formel Eintrag für Eintrag zuführen, wenn man am Schluß einer Schleife jeweils das Adreßregister, um die Anzahl der Bytes erhöht, die ein Koordinatenpaar benötigt, also um 12 Bytes. Mit MULS, DIVS läßt sich die Rechnung in 68008 direkt durchführen.

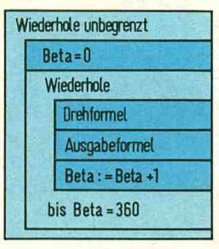


Bild 7. Das Struktogramm Drehung

Die Instruktion EXT.L ist vor einer Division nötig, da die Division mit 32 Bit als Divisor arbeitet.
Das Unterprogramm MOVETO legt den Startpunkt auf dem Bildschirm fest, mit DRAWTO kann man eine Linie zeichnen. Die Koordinate x muß im Register D1 und die Koordinate y im Register D2 stehen.

Der Würfel wird gedreht

Das Programm ist schon ganz praktisch, aber es wäre sicher sehr hübsch, einen sich drehenden Würfel zu zeigen.
Dazu muß man die Koordinaten des Würfels vor seiner Projektion mit Hilfe einer weiteren mathematischen Formel umrechnen. Es gibt eine praktische Drehformel, die wie folgt lautet:

$$X' = X * COS(BETA) + Z * SIN(BETA)$$

$$Y' = Y$$

$$Z' = -X * SIN(BETA) + Z * COS(BETA)$$

Diese Formel dreht jeden Punkt mit den Koordinaten x, y, z um den Winkel BETA um die y-Achse. Daher ändert sich der Wert für y auch nicht. Wenn man den Würfel um eine andere Achse drehen will, so gibt es entsprechende Formeln, wenn man x,y,z zyklisch vertauscht. Will man einen Gegenstand frei im Raum drehen, so kann man ihn zunächst in den Ursprung bewegen, dann die Drehformel anwenden und ihn dann zurücktransportieren. *Bild 7* zeigt das Struktogramm. In einer inneren Schleife wird jeweils die Drehformel angewen-

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1


009C00                  ;************************
009C00                  ; AUSFLUG NACH SIMLAND  ;
009C00                  ; VERSION 1.0  RDK      ;
009C00                  ; MIT PERSPEKTIVE       ;
009C00                  ;  MIT DREHUNG          ;
009C00                  ;************************
009C00
00A000                  ORG $A000
00A000
00A000                  ; ***** VEREINBARUNGS-BAUSTEIN *****
00A000
00A000                  ; WUERFEL - VERBINDUNGSLINIEN
00A000
00A000                  WUERFEL:
00A000  FF9C FF9C FF9C  DC.W -100,-100,-100,100,-100,-100
00A006  0064 FF9C FF9C
00A00C  FF9C FF9C FF9C  DC.W -100,-100,-100,-100,-100,100
00A012  FF9C FF9C 0064
00A018  FF9C FF9C 0064  DC.W -100,-100,100,100,-100,100
00A01E  0064 FF9C 0064
00A024  0064 FF9C 0064  DC.W 100,-100,100,100,-100,-100
00A02A  0064 FF9C FF9C
00A030  0064 FF9C FF9C  DC.W 100,-100,-100,100,100,-100
00A036  0064 0064 FF9C
00A03C  FF9C FF9C FF9C  DC.W -100,-100,-100,-100,100,-100
00A042  FF9C 0064 FF9C
00A048  FF9C FF9C 0064  DC.W -100,-100,100,-100,100,100
00A04E  FF9C 0064 0064
00A054  0064 FF9C 0064  DC.W 100,-100,100,100,100,100
00A05A  0064 0064 0064
00A060  FF9C 0064 FF9C  DC.W -100,100,-100,100,100,-100
00A066  0064 0064 FF9C
00A06C  FF9C 0064 FF9C  DC.W -100,100,-100,-100,100,100
00A072  FF9C 0064 0064
00A078  FF9C 0064 0064  DC.W -100,100,100,100,100,100
00A07E  0064 0064 0064
00A084  0064 0064 0064  DC.W 100,100,100,100,100,-100
00A08A  0064 0064 FF9C
00A090
= 0000000C              ANZAHL EQU 12 ; ANZAHL KANTEN
= 0000000C              KOOR   EQU 12 ; ANZAHL BYTES PRO EINTRAG
00A090
= 00000000              X1 EQU 0  ; OFFSET-ADRESSIERUNG
= 00000002              Y1 EQU 2  ; WORTDISTANZ
= 00000004              Z1 EQU 4
= 00000006              X2 EQU 6
= 00000008              Y2 EQU 8
= 0000000A              Z2 EQU 10
```

Bild 8. Das Programm Drehung

```
00A090
= FFFFFC18              Y0 EQU -1000 ; AUGPUNKT
= 000001F4              PERSP EQU 500 ; AUGHOEHE
00A090
00A090  0000            XF1: DC.W 0  ; VARIABLE
00A092  0000            ZF1: DC.W 0
00A094  0000            XF2: DC.W 0
00A096  0000            ZF2: DC.W 0
00A098
00A098  0000            X11: DC.W  0
00A09A  0000            Z11: DC.W  0
00A09C
00A09C  0000            X22: DC.W  0
00A09E  0000            Z22: DC.W  0
00A0A0
00A0A0  0000            BETHA: DC.W  0
00A0A2  0000            SINB: DC.W  0 ; BLEIBEN CONSTANT INNERHALB
00A0A4  0000            COSB: DC.W  0 ; WDH1
00A0A6
00A0A6                  ; ***** PROGRAMM-BAUSTEIN *****
00A0A6
00A0A6                  START:
00A0A6  4279 0000A0A0    CLR BETHA       ; STARTWINKEL
00A0AC                  WDH0:
00A0AC  41F9 0000A000    LEA WUERFEL,A0  ; STARTADRESSE
00A0B2  3039 0000A0A0    MOVE BETHA,D0   ;NUR EINMAL BERECHNEN PRO SCHLEIFE
00A0B8  4EB9 00001034    JSR @SIN
00A0BE  33C0 0000A0A2    MOVE D0,SINB
00A0C4  3039 0000A0A0    MOVE BETHA,D0
00A0CA  4EB9 00001030    JSR @COS
00A0D0  33C0 0000A0A4    MOVE D0,COSB
00A0D6                  ;
00A0D6  3A3C 000B        MOVE #ANZAHL-1,D5 ; SCHLEIFENZAEHLER
00A0DA                  WDH1:
00A0DA                  ; DREHFORMEL
00A0DA  3028 0000        MOVE X1(A0),D0
00A0DE  C1F9 0000A0A4    MULS COSB,D0
00A0E4  3228 0004        MOVE Z1(A0),D1
00A0E8  C3F9 0000A0A2    MULS SINB,D1
00A0EE  D081             ADD.L D1,D0
00A0F0  81FC 0100        DIVS #256,D0 ; NORMIEREN
00A0F4  33C0 0000A098    MOVE D0,X11  ; X':=x*cos(b)+z*sin(b)
00A0FA  3028 0000        MOVE X1(A0),D0
00A0FE  4440             NEG D0
00A100  C1F9 0000A0A2    MULS SINB,D0
00A106  3228 0004        MOVE Z1(A0),D1
00A10A  C3F9 0000A0A4    MULS COSB,D1
00A110  D081             ADD.L D1,D0
00A112  81FC 0100        DIVS #256,D0 ; NORMIEREN
00A116  33C0 0000A09A    MOVE D0,Z11  ; Z':=-x*sin(b)+z*cos(b)
00A11C  3028 0006        MOVE X2(A0),D0
00A120  C1F9 0000A0A4    MULS COSB,D0
00A126  3228 000A        MOVE Z2(A0),D1
00A12A  C3F9 0000A0A2    MULS SINB,D1
00A130  D081             ADD.L D1,D0
00A132  81FC 0100        DIVS #256,D0 ; NORMIEREN
00A136  33C0 0000A09C    MOVE D0,X22
00A13C  3028 0006        MOVE X2(A0),D0
00A140  4440             NEG D0
00A142  C1F9 0000A0A2    MULS SINB,D0
00A148  3228 000A        MOVE Z2(A0),D1
00A14C  C3F9 0000A0A4    MULS COSB,D1
00A152  D081             ADD.L D1,D0
00A154  81FC 0100        DIVS #256,D0 ; NORMIEREN
00A158  33C0 0000A09E    MOVE D0,Z22
00A15E                  ; AUGABETEIL
00A15E  3039 0000A098    MOVE X11,D0
00A164  C1FC FC18        MULS #Y0,D0
00A168  323C FC18        MOVE #Y0,D1
00A16C  9268 0002        SUB Y1(A0),D1
00A170  81C1             DIVS D1,D0 ; XF1 BERECHNET
00A172  33C0 0000A090    MOVE D0,XF1
00A178  3039 0000A09A    MOVE Z11,D0
00A17E  0640 01F4        ADD #PERSP,D0
00A182  C1FC FC18        MULS #Y0,D0
00A186  81C1             DIVS D1,D0 ; ZF1 BERECHNET
00A188  33C0 0000A092    MOVE D0,ZF1
00A18E  3039 0000A09C    MOVE X22,D0
00A194  C1FC FC18        MULS #Y0,D0
00A198  323C FC18        MOVE #Y0,D1
00A19C  9268 0008        SUB Y2(A0),D1
00A1A0  81C1             DIVS D1,D0
00A1A2  33C0 0000A094    MOVE D0,XF2 ; XF1 BERECHNET
00A1A8  3039 0000A09E    MOVE Z22,D0
00A1AE  0640 01F4        ADD #PERSP,D0
00A1B2  C1FC FC18        MULS #Y0,D0
00A1B6  81C1             DIVS D1,D0
00A1B8  33C0 0000A096    MOVE D0,ZF2
00A1BE  3239 0000A090    MOVE XF1,D1
00A1C4  3439 0000A092    MOVE ZF1,D2
00A1CA  0641 0100        ADD #256,D1  ; IN BILDSCHIRMITTE BRINGEN
00A1CE  0642 0100        ADD #256,D2
00A1D2  0442 01F4        SUB #PERSP,D2
00A1D6  48C2             EXT.L D2
00A1D8  85FC 0002        DIVS #2,D2
00A1DC  4EB9 00000EE2    JSR @MOVETO
00A1E2  3239 0000A094    MOVE XF2,D1
00A1E8  3439 0000A096    MOVE ZF2,D2
00A1EE  0641 0100        ADD #256,D1 ; IN BILDSCHIRMMITTE BRINGEN
00A1F2  0442 01F4        SUB #PERSP,D2
00A1F6  0642 0100        ADD #256,D2
00A1FA  48C2             EXT.L D2
00A1FC  85FC 0002        DIVS #2,D2
00A200  4EB9 00000F4C    JSR @DRAWTO
00A206
00A206  D1FC 0000000C    ADDA.L #KOOR,A0
00A20C  51CD FECC        DBRA D5,WDH1 ; BIS ALLE KANTEN BERECHNET
00A210
00A210  0679 0001        ADD #1,BETHA
00A214  0000A0A0
00A218  3039 0000A0A0    MOVE BETHA,D0
00A21E  0C40 0168        CMP #360,D0
00A222  6600 FE88        BNE WDH0
00A226
00A226  4E75            RTS
00A228
00A228
00A228                  ; ********
00A228                   END
008C10  Ende-Symboltabelle
```

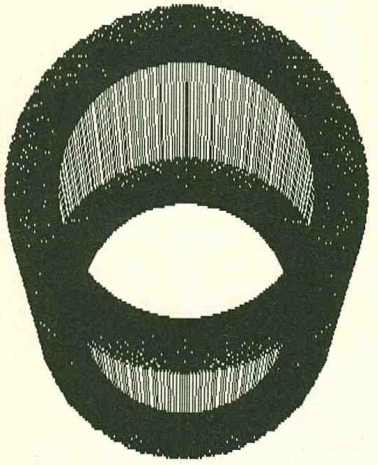


Bild 9. Das Ergebnis zu Bild 8

F=Flip M=Menue

det, dann der gedrehte Würfel ausgegeben und schließlich der Winkel um 1 erhöht. Man kann das Programm auch anders gestalten, denn bei dem einfachen Verfahren hier muß man sich alle Koordinaten, die durch die Drehformel berechnet wurden, in einem getrennten Speicher aufbewahren. Man könnte zunächst die Eckpunkte einer Kante drehen, dann projezieren, dann die nächste Kante bearbeiten und so weiter.

Bild 8 zeigt die programmtechnische Lösung. SIN und COS des Winkels müssen nur einmal zu Beginn der Schleife berechnet werden, denn sie bleiben für einen gesamten Berechnungsvorgang konstant. Für SIN und COS gibt es Unterprogramme im Grundprogramm, die den Wert berechnen, allerdings mit der Formel 256∗SIN und 256∗COS, und die Nachkommastellen werden abgeschnitten. Die Eingabe des Winkels muß im Register D0 in Grad erfolgen. Wenn man das Programm startet, ergibt sich ein Bild nach *Bild 9*: Der Würfel ist nicht mehr erkennbar. Bevor man einen neuen Würfel ausgeben kann, muß man den alten löschen, sonst sieht es wie in Bild 9 aus.

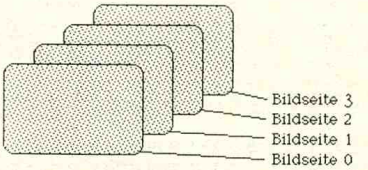


Bild 10. Die Bildseiten der GDP-Baugruppe

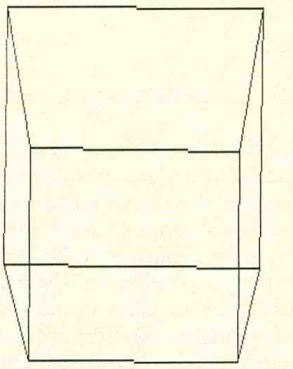


Bild 12. Nach dem Programmlauf

F=Flip M=Menue

Man füge dazu hinter die Marke WDH0 ein kleines Programmstück ein:

```
        ...
WDH0:
        JSR @SYNC    ; warten auf
                       20-ms-Signal
        BEQ WDH0
        MOVE #4,D0   ; Löschbefehl
                       für den GDP
        JSR @CMD     ; direkt ausgeben

        ...
```

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

009C00                  ;************************
009C00                  ; AUSFLUG NACH SIMLAND  ;
009C00                  ; VERSION 1.0  RDK      ;
009C00                  ; MIT PERSPEKTIVE       ;
009C00                  ;  MIT DREHUNG          ;
009C00                  ; MIT KORREKTUR         ;
009C00                  ;************************
009C00
00B000                  ORG $B000
00B000
00B000                  ; ***** VEREINBARUNGS-BAUSTEIN *****
00B000
00B000                  ; WUERFEL - VERBINDUNGSLINIEN
00B000
00B000                  WUERFEL:
00B000  FF9C FF9C FF9C  DC.W -100,-100,-100,100,-100,-100
00B006  0064 FF9C FF9C
00B00C  FF9C FF9C FF9C  DC.W -100,-100,-100,-100,-100,100
00B012  FF9C FF9C 0064
00B018  FF9C FF9C 0064  DC.W -100,-100,100,100,-100,100
00B01E  0064 FF9C 0064
00B024  0064 FF9C 0064  DC.W 100,-100,100,100,-100,-100
00B02A  0064 FF9C FF9C
00B030  0064 FF9C FF9C  DC.W 100,-100,-100,100,100,-100
00B036  0064 0064 FF9C
00B03C  FF9C FF9C FF9C  DC.W -100,-100,-100,-100,100,-100
00B042  FF9C 0064 FF9C
00B048  FF9C FF9C 0064  DC.W -100,-100,100,-100,100,100
00B04E  FF9C 0064 0064
00B054  0064 FF9C 0064  DC.W 100,-100,100,100,100,100
00B05A  0064 0064 0064
00B060  FF9C 0064 FF9C  DC.W -100,100,-100,100,100,-100
00B066  0064 0064 FF9C
00B06C  FF9C 0064 FF9C  DC.W -100,100,-100,-100,100,100
00B072  FF9C 0064 0064
00B078  FF9C 0064 0064  DC.W -100,100,100,100,100,100
00B07E  0064 0064 0064
00B084  0064 0064 0064  DC.W 100,100,100,100,100,-100
00B08A  0064 0064 FF9C
00B090
= 0000000C              ANZAHL EQU 12 ; ANZAHL KANTEN
= 0000000C              KOOR   EQU 12 ; ANZAHL BYTES PRO EINTRAG
00B090
= 00000000              X1 EQU 0  ; OFFSET-ADRESSIERUNG
= 00000002              Y1 EQU 2  ; WORTDISTANZ
= 00000004              Z1 EQU 4
= 00000006              X2 EQU 6
= 00000008              Y2 EQU 8
= 0000000A              Z2 EQU 10
00B090
= FFFFFC18              Y0 EQU -1000 ; AUGPUNKT
= 000001F4              PERSP EQU 500 ; AUGHOEHE
00B090
00B090  0000            XF1: DC.W 0  ; VARIABLE
00B092  0000            ZF1: DC.W 0
00B094  0000            XF2: DC.W 0
00B096  0000            ZF2: DC.W 0
00B098
00B098  0000            X11: DC.W  0
00B09A  0000            Z11: DC.W  0
00B09C
00B09C  0000            X22: DC.W  0
00B09E  0000            Z22: DC.W  0
00B0A0
00B0A0  0000            BETHA: DC.W  0
00B0A2  0000            SINB: DC.W  0 ; BLEIBEN CONSTANT INNERHALB
00B0A4  0000            COSB: DC.W  0 ; WDH1
00B0A6
00B0A6  00              LESE: DC.B 0 ; FUER BILDSEITEN
00B0A7  00              SCHREIB: DC.B 0
00B0A8
```

Bild 11. Drehung mit zwei Bildseiten realisiert

```
00B0A8
00B0A8                  ; ***** UNTERPROGRAMM-BAUSTEIN *****
00B0A8
00B0A8                  GIBAUS:
00B0A8  41F9 0000B000    LEA WUERFEL,A0  ; STARTADRESSE
00B0AE  3039 0000B0A0    MOVE BETHA,D0   ;NUR EINMAL BERECHNEN PRO SCHLEIFE
00B0B4  4EB9 00001034    JSR @SIN
00B0BA  33C0 0000B0A2    MOVE D0,SINB
00B0C0  3039 0000B0A0    MOVE BETHA,D0
00B0C6  4EB9 00001030    JSR @COS
00B0CC  33C0 0000B0A4    MOVE D0,COSB
00B0D2                  ;
00B0D2  3A3C 000B        MOVE #ANZAHL-1,D5 ; SCHLEIFENZAEHLER
00B0D6                  WDH1:
00B0D6                  ; DREHFORMEL
00B0D6  3028 0000        MOVE X1(A0),D0
00B0DA  C1F9 0000B0A4    MULS COSB,D0
00B0E0  3228 0004        MOVE Z1(A0),D1
00B0E4  C3F9 0000B0A2    MULS SINB,D1
00B0EA  D081             ADD.L D1,D0
00B0EC  81FC 0100        DIVS #256,D0 ; NORMIEREN
00B0F0  33C0 0000B098    MOVE D0,X11  ; X':=x*cos(b)+z*sin(b)
00B0F6  3028 0000        MOVE X1(A0),D0
00B0FA  4440             NEG D0
00B0FC  C1F9 0000B0A2    MULS SINB,D0
00B102  3228 0004        MOVE Z1(A0),D1
00B106  C3F9 0000B0A4    MULS COSB,D1
00B10C  D081             ADD.L D1,D0
00B10E  81FC 0100        DIVS #256,D0 ; NORMIEREN
00B112  33C0 0000B09A    MOVE D0,Z11  ; Z':=-x*sin(b)+z*cos(b)
00B118  3028 0006        MOVE X2(A0),D0
00B11C  C1F9 0000B0A4    MULS COSB,D0
00B122  3228 000A        MOVE Z2(A0),D1
00B126  C3F9 0000B0A2    MULS SINB,D1
00B12C  D081             ADD.L D1,D0
00B12E  81FC 0100        DIVS #256,D0 ; NORMIEREN
00B132  33C0 0000B09C    MOVE D0,X22
00B138  3028 0006        MOVE X2(A0),D0
00B13C  4440             NEG D0
00B13E  C1F9 0000B0A2    MULS SINB,D0
00B144  3228 000A        MOVE Z2(A0),D1
00B148  C3F9 0000B0A4    MULS COSB,D1
00B14E  D081             ADD.L D1,D0
00B150  81FC 0100        DIVS #256,D0 ; NORMIEREN
00B154  33C0 0000B09E    MOVE D0,Z22
00B15A                  ; AUSGABETEIL
00B15A  3039 0000B098    MOVE X11,D0
00B160  C1FC FC18        MULS #Y0,D0
00B164  323C FC18        MOVE #Y0,D1
00B168  9268 0002        SUB Y1(A0),D1
00B16C  81C1             DIVS D1,D0 ; XF1 BERECHNET
00B16E  33C0 0000B090    MOVE D0,XF1
00B174  3039 0000B09A    MOVE Z11,D0
00B17A  0640 01F4        ADD #PERSP,D0
00B17E  C1FC FC18        MULS #Y0,D0
00B182  81C1             DIVS D1,D0 ; ZF1 BERECHNET
00B184  33C0 0000B092    MOVE D0,ZF1
00B18A  3039 0000B09C    MOVE X22,D0
00B190  C1FC FC18        MULS #Y0,D0
00B194  323C FC18        MOVE #Y0,D1
00B198  9268 0008        SUB Y2(A0),D1
00B19C  81C1             DIVS D1,D0
00B19E  33C0 0000B094    MOVE D0,XF2 ; XF1 BERECHNET
00B1A4  3039 0000B09E    MOVE Z22,D0
00B1AA  0640 01F4        ADD #PERSP,D0
00B1AE  C1FC FC18        MULS #Y0,D0
00B1B2  81C1             DIVS D1,D0
00B1B4  33C0 0000B096    MOVE D0,ZF2
00B1BA  3239 0000B090    MOVE XF1,D1
00B1C0  3439 0000B092    MOVE ZF1,D2
00B1C6  0641 0100        ADD #256,D1  ; IN BILDSCHIRMITTE BRINGEN
00B1CA  0642 0100        ADD #256,D2
00B1CE  0442 01F4        SUB #PERSP,D2
00B1D2  48C2             EXT.L D2
00B1D4  85FC 0002        DIVS #2,D2
00B1D8  4EB9 00000EE2    JSR @MOVETO
00B1DE  3239 0000B094    MOVE XF2,D1
00B1E4  3439 0000B096    MOVE ZF2,D2
00B1EA  0641 0100        ADD #256,D1 ; IN BILDSCHIRMMITTE BRINGEN
00B1EE  0442 01F4        SUB #PERSP,D2
00B1F2  0642 0100        ADD #256,D2
00B1F6  48C2             EXT.L D2
00B1F8  85FC 0002        DIVS #2,D2
00B1FC  4EB9 00000F4C    JSR @DRAWTO
00B202
00B202  D1FC 0000000C    ADDA.L #KOOR,A0
00B208  51CD FECC        DBRA D5,WDH1 ; BIS ALLE KANTEN BERECHNET
00B20C  4E75            RTS
00B20E
00B20E
00B20E                  ; ****** PROGRAMM-BAUSTEIN *****
00B20E
00B20E                  START:
00B20E  4279 0000B0A0    CLR BETHA
00B214  13FC 0000        MOVE.B #0,LESE
00B218  0000B0A6
00B21C  13FC 0001        MOVE.B #1,SCHREIB
00B220  0000B0A7
00B224                  WDH0:
00B224  1239 0000B0A6    MOVE.B LESE,D1
00B22A  1039 0000B0A7    MOVE.B SCHREIB,D0
00B230  4EB9 00000E0C    JSR @NEWPAGE  ; BILDSEITE ANWAEHLEN
00B236  4EB9 00000D6E    JSR @ERAPEN ; ZUERST LOESCHEN MIT ALTEM WINKEL
00B23C  4EB9 0000B0A8    JSR GIBAUS
00B242  0679 0002        ADD #2,BETHA ; DANN AUSGEBEN MIT NEUEM
00B246  0000B0A0
00B24A  4EB9 00000D7C    JSR @SETPEN
00B250  4EB9 0000B0A8    JSR GIBAUS
00B256  0479 0001        SUB #1,BETHA ; N-1
00B25A  0000B0A0
00B25E  0A39 0001        EORI.B #1,LESE
00B262  0000B0A6
00B266  0A39 0001        EORI.B #1,SCHREIB
00B26A  0000B0A7
00B26E  3039 0000B0A0    MOVE BETHA,D0
00B274  0C40 0168        CMP #360,D0
00B278  6600 FFAA        BNE WDH0
00B27C
00B27C  4E75            RTS
00B27E
00B27E                  ; ********
00B27E                   END

008C46  Ende-Symboltabelle
```

```
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1
009C00                  ;************************
009C00                  ; AUSFLUG NACH SIMLAND  ;
009C00                  ; VERSION 1.0  RDK      ;
009C00                  ; MIT PERSPEKTIVE       ;
009C00                  ;  MIT DREHUNG          ;
009C00                  ; MIT KORREKTUR         ;
009C00                  ; MIT TASTATUR BEWEGUNG ;
009C00                  ;************************
00B000                  ORG $B000
00B000                  ; ***** VEREINBARUNGS-BAUSTEIN *****
= FFFFFF30              TASTEN EQU $FFFFFF30
00B000
00B000                  ; WUERFEL - VERBINDUNGSLINIEN
00B000
00B000                  WUERFEL:
00B000  FF9C FF9C FF9C  DC.W -100,-100,-100,100,-100,-100
00B006  0064 FF9C FF9C
00B00C  FF9C FF9C FF9C  DC.W -100,-100,-100,-100,-100,100
00B012  FF9C FF9C 0064
00B018  FF9C FF9C 0064  DC.W -100,-100,100,100,-100,100
00B01E  0064 FF9C 0064
00B024  0064 FF9C 0064  DC.W 100,-100,100,100,-100,-100
00B02A  0064 FF9C FF9C
00B030  0064 FF9C FF9C  DC.W 100,-100,-100,100,100,-100
00B036  0064 0064 FF9C
00B03C  FF9C FF9C FF9C  DC.W -100,-100,-100,-100,100,-100
00B042  FF9C 0064 FF9C
00B048  FF9C FF9C 0064  DC.W -100,-100,100,-100,100,100
00B04E  FF9C 0064 0064
00B054  0064 FF9C 0064  DC.W 100,-100,100,100,100,100
00B05A  0064 0064 0064
00B060  FF9C 0064 FF9C  DC.W -100,100,-100,100,100,-100
00B066  0064 0064 FF9C
00B06C  FF9C 0064 FF9C  DC.W -100,100,-100,-100,100,100
00B072  FF9C 0064 0064
00B078  FF9C 0064 0064  DC.W -100,100,100,100,100,100
00B07E  0064 0064 0064
00B084  0064 0064 0064  DC.W 100,100,100,100,100,-100
00B08A  0064 0064 FF9C
00B090
= 0000000C              ANZAHL EQU 12 ; ANZAHL KANTEN
= 0000000C              KOOR   EQU 12 ; ANZAHL BYTES PRO EINTRAG
00B090
= 00000000              X1 EQU 0  ; OFFSET-ADRESSIERUNG
= 00000002              Y1 EQU 2  ; WORTDISTANZ
= 00000004              Z1 EQU 4
= 00000006              X2 EQU 6
= 00000008              Y2 EQU 8
= 0000000A              Z2 EQU 10
00B090
00B090
00B090  0000            Y0: DC.W  0 ;AUCH VARIABEL, AUSPUNKT
00B092  0000            Y0A: DC.W 0
00B094  0000            Y0B: DC.W  0
00B096  0000            Y0C: DC.W  0
00B098
00B098  0000            PERSP: DC.W  0
00B09A  0000            PERSPA: DC.W 0
00B09C  0000            PERSPB: DC.W 0
00B09E  0000            PERSPC: DC.W  0
```

Bild 13. Ein Programm mit Tastensteuerung

```
00B0A0
00B0A0  0000            XF1: DC.W 0  ; VARIABLE
00B0A2  0000            ZF1: DC.W 0
00B0A4  0000            XF2: DC.W 0
00B0A6  0000            ZF2: DC.W 0
00B0AB
00B0A8  0000            X11: DC.W  0
00B0AA  0000            Z11: DC.W  0
00B0AC
00B0AC  0000            X22: DC.W  0
00B0AE  0000            Z22: DC.W  0
00B0B0
00B0B0  0000            BETHA: DC.W  0
00B0B2  0000            BETHAA: DC.W  0
00B0B4  0000            BETHAB: DC.W  0
00B0B6  0000            BETHAC: DC.W  0
00B0B8
00B0B8  0000            SINB: DC.W  0 ; BLEIBEN CONSTANT INNERHALB
00B0BA  0000            COSB: DC.W  0 ; WDH1
00B0BC
00B0BC  00              LESE: DC.B 0 ; FUER BILDSEITEN
00B0BD  00              SCHREIB: DC.B 0
00B0BE
00B0BE
00B0BE                  ; ***** UNTERPROGRAMM-BAUSTEIN *****
00B0BE
00B0BE                  GIBAUS:
00B0BE  41F9 0000B000    LEA WUERFEL,A0  ; STARTADRESSE
00B0C4  3039 0000B0B0    MOVE BETHA,D0   ;NUR EINMAL BERECHNEN PRO SCHLEIFE
00B0CA  4EB9 00001034    JSR @SIN
00B0D0  33C0 0000B0B8    MOVE D0,SINB
00B0D6  3039 0000B0B0    MOVE BETHA,D0
00B0DC  4EB9 00001030    JSR @COS
00B0E2  33C0 0000B0BA    MOVE D0,COSB
00B0E8                  ;
00B0E8  3A3C 000B        MOVE #ANZAHL-1,D5 ; SCHLEIFENZAEHLER
00B0EC                  WDH1:
00B0EC                  ; DREHFORMEL
00B0EC  3028 0000        MOVE X1(A0),D0
00B0F0  C1F9 0000B0BA    MULS COSB,D0
00B0F6  3228 0004        MOVE Z1(A0),D1
00B0FA  C3F9 0000B0B8    MULS SINB,D1
00B100  D081             ADD.L D1,D0
00B102  81FC 0100        DIVS #256,D0 ; NORMIEREN
00B106  33C0 0000B0A8    MOVE D0,X11  ; X':=x*cos(b)+z*sin(b)
00B10C  3028 0000        MOVE X1(A0),D0
00B110  4440             NEG D0
00B112  C1F9 0000B0B8    MULS SINB,D0
00B118  3228 0004        MOVE Z1(A0),D1
00B11C  C3F9 0000B0BA    MULS COSB,D1
00B122  D081             ADD.L D1,D0
00B124  81FC 0100        DIVS #256,D0 ; NORMIEREN
00B128  33C0 0000B0AA    MOVE D0,Z11  ; Z':=-x*sin(b)+z*cos(b)
00B12E  3028 0006        MOVE X2(A0),D0
00B132  C1F9 0000B0BA    MULS COSB,D0
00B138  3228 000A        MOVE Z2(A0),D1
00B13C  C3F9 0000B0B8    MULS SINB,D1
00B142  D081             ADD.L D1,D0
00B144  81FC 0100        DIVS #256,D0 ; NORMIEREN
00B148  33C0 0000B0AC    MOVE D0,X22
00B14E  3028 0006        MOVE X2(A0),D0
00B152  4440             NEG D0
00B154  C1F9 0000B0B8    MULS SINB,D0
00B15A  3228 000A        MOVE Z2(A0),D1
```

```
00B15E  C3F9 0000B0BA     MULS COSB,D1
00B164  D081              ADD.L D1,D0
00B166  81FC 0100         DIVS #256,D0 ; NORMIEREN
00B16A  33C0 0000B0AE     MOVE D0,Z22
00B170                   ; AUSGABETEIL
00B170  3039 0000B0A8     MOVE X11,D0
00B176  C1F9 0000B090     MULS Y0,D0
00B17C  3239 0000B090     MOVE Y0,D1
00B182  9268 0002         SUB Y1(A0),D1
00B186  81C1              DIVS D1,D0 ; XF1 BERECHNET
00B188  33C0 0000B0A0     MOVE D0,XF1
00B18E  3039 0000B0AA     MOVE Z11,D0
00B194  D079 0000B098     ADD PERSP,D0
00B19A  C1F9 0000B090     MULS Y0,D0
00B1A0  81C1              DIVS D1,D0 ; ZF1 BERECHNET
00B1A2  33C0 0000B0A2     MOVE D0,ZF1
00B1A8  3039 0000B0AC     MOVE X22,D0
00B1AE  C1F9 0000B090     MULS Y0,D0
00B1B4  3239 0000B090     MOVE Y0,D1
00B1BA  9268 0008         SUB Y2(A0),D1
00B1BE  81C1              DIVS D1,D0
00B1C0  33C0 0000B0A4     MOVE D0,XF2 ; XF1 BERECHNET
00B1C6  3039 0000B0AE     MOVE Z22,D0
00B1CC  D079 0000B098     ADD PERSP,D0
00B1D2  C1F9 0000B090     MULS Y0,D0
00B1D8  81C1              DIVS D1,D0
00B1DA  33C0 0000B0A6     MOVE D0,ZF2
00B1E0  3239 0000B0A0     MOVE XF1,D1
00B1E6  3439 0000B0A2     MOVE ZF1,D2
00B1EC  0641 0100         ADD #256,D1  ; IN BILDSCHIRMITTE BRINGEN
00B1F0  0642 0100         ADD #256,D2
00B1F4  9479 0000B098     SUB PERSP,D2
00B1FA  48C2              EXT.L D2
00B1FC  85FC 0002         DIVS #2,D2
00B200  4EB9 00000EE2     JSR @MOVETO
00B206  3239 0000B0A4     MOVE XF2,D1
00B20C  3439 0000B0A6     MOVE ZF2,D2
00B212  0641 0100         ADD #256,D1 ; IN BILDSCHIRMMITTE BRINGEN
00B216  9479 0000B098     SUB PERSP,D2
00B21C  0642 0100         ADD #256,D2
00B220  48C2              EXT.L D2
00B222  85FC 0002         DIVS #2,D2
00B226  4EB9 00000F4C     JSR @DRAWTO
00B22C
00B22C  D1FC 0000000C     ADDA.L #KOOR,A0
00B232  51CD FEB8         DBRA D5,WDH1 ; BIS ALLE KANTEN BERECHNET
00B236  4E75              RTS
00B238
00B238
00B238                   ; ****** PROGRAMM-BAUSTEIN *****
00B238
00B238                   START:
00B238  33FC FE0C         MOVE #-500,Y0
00B23C  0000B090
00B240  33FC FE0C         MOVE #-500,Y0A
00B244  0000B092
00B248  33FC FE0C         MOVE #-500,Y0B
00B24C  0000B094
00B250  33FC FE0C         MOVE #-500,Y0C
00B254  0000B096
00B258  33FC 00C8         MOVE #200,PERSP
00B25C  0000B098
00B260  33FC 00C8         MOVE #200,PERSPA
00B264  0000B09A
00B268  33FC 00C8         MOVE #200,PERSPB
00B26C  0000B09C
00B270  33FC 00C8         MOVE #200,PERSPC
00B274  0000B09E
00B278                   ;
00B278  4279 0000B0B0     CLR BETHA
00B27E  4279 0000B0B2     CLR BETHAA
00B284  4279 0000B0B4     CLR BETHAB
00B28A  4279 0000B0B6     CLR BETHAC
00B290  13FC 0000         MOVE.B #0,LESE
00B294  0000B0BC
00B298  13FC 0001         MOVE.B #1,SCHREIB
00B29C  0000B0BD
00B2A0                   WDH0:
00B2A0  1239 0000B0BC     MOVE.B LESE,D1
00B2A6  1039 0000B0BD     MOVE.B SCHREIB,D0
00B2AC  4EB9 00000E0C     JSR @NEWPAGE  ; BILDSEITE ANWAEHLEN
00B2B2  4EB9 00000D6E     JSR @ERAPEN ; ZUERST LOESCHEN MIT ALTEM WINKEL
00B2B8  33F9 0000B0B6     MOVE BETHAC,BETHA
00B2BE  0000B0B0
00B2C2  33F9 0000B096     MOVE Y0C,Y0
00B2C8  0000B090
00B2CC  33F9 0000B09E     MOVE PERSPC,PERSP
00B2D2  0000B098
00B2D6  4EB9 0000B0BE     JSR GIBAUS
00B2DC  0679 0002         ADD #2,BETHA
00B2E0                   ; DANN AUSGEBEN MIT NEUEM
00B2E0  0000B0B0
00B2E4  4EB9 00000D7C     JSR @SETPEN
00B2EA  33F9 0000B0B2     MOVE BETHAA,BETHA
00B2F0  0000B0B0
00B2F4  33F9 0000B092     MOVE Y0A,Y0
00B2FA  0000B090
00B2FE  33F9 0000B09A     MOVE PERSPA,PERSP
00B304  0000B098
00B308  4EB9 0000B0BE     JSR GIBAUS
00B30E  33F9 0000B0B4     MOVE BETHAB,BETHAC
00B314  0000B0B6
00B318  33F9 0000B0B2     MOVE BETHAA,BETHAB
00B31E  0000B0B4
00B322  33F9 0000B094     MOVE Y0B,Y0C
00B328  0000B096
00B32C  33F9 0000B092     MOVE Y0A,Y0B
00B332  0000B094
00B336  33F9 0000B09C     MOVE PERSPB,PERSPC
00B33C  0000B09E
00B340  33F9 0000B09A     MOVE PERSPA,PERSPB
00B346  0000B09C
00B34A  0839 0000         BTST.B #0,TASTEN
00B34E  FFFFFF30
00B352  6600 000A         BNE SK1
00B356  0679 0005         ADD #5,BETHAA
00B35A  0000B0B2
00B35E
00B35E                   SK1:
00B35E  0839 0001         BTST.B #1,TASTEN
00B362  FFFFFF30
00B366  6600 000A         BNE SK2
00B36A  0479 0005         SUB #5,BETHAA
00B36E  0000B0B2
00B372
00B372                   SK2:
00B372  0839 0002         BTST.B #2,TASTEN
00B376  FFFFFF30
00B37A  6600 000A         BNE SK3
00B37E  0679 0005         ADD #5,Y0A
00B382  0000B092
00B386
00B386                   SK3:
00B386  0839 0003         BTST.B #3,TASTEN
00B38A  FFFFFF30
00B38E  6600 000A         BNE SK4
00B392  0479 0005         SUB #5,Y0A
00B396  0000B092
00B39A
00B39A                   SK4:
00B39A  0839 0004         BTST.B #4,TASTEN
00B39E  FFFFFF30
00B3A2  6600 000A         BNE SK5
00B3A6  0679 0005         ADD #5,PERSPA
00B3AA  0000B09A
00B3AE
00B3AE                   SK5:
00B3AE  0839 0005         BTST.B #5,TASTEN
00B3B2  FFFFFF30
00B3B6  6600 000A         BNE FINAL
00B3BA  0479 0005         SUB #5,PERSPA
00B3BE  0000B09A
00B3C2
00B3C2                   FINAL:
00B3C2
00B3C2  0A39 0001         EORI.B #1,LESE
00B3C6  0000B0BC
00B3CA  0A39 0001         EORI.B #1,SCHREIB
00B3CE  0000B0BD
00B3D2  6000 FECC         BRA WDH0
00B3D6
00B3D6
00B3D6                   ; ********
00B3D6                    END

008D66  Ende-Symboltabelle
```

Mit CMD kann man einen Befehl direkt an den Grafikprozessor ausgeben. Der Befehl mit dem Code 4 löscht die aktuelle Bildseite innerhalb von 20 ms. Daher sollte man zusätzlich SYNC anwenden, denn sonst sieht man den Würfel nicht mehr, da sonst mehr gelöscht als geschrieben wird (ausprobieren).

Jetzt wird gefeilt

Das Programm ist aber noch nicht sehr elegant. Störend ist das Flimmern durch den Löschvorgang. Eine bessere Lösung ist die Verwendung von mehreren Bildebenen zum Zeichnen. Auf der GDP64-Baugruppe sind vier Bildseiten untergebracht, von denen aber immer nur eine dargestellt werden kann. *Bild 10* zeigt ein Schema. Welche Bildseite dargestellt wird und auf welche geschrieben wird, das kann man an einem Port des GDPs einstellen. Man kann also sogar eine Bildseite darstellen und auf einer anderen in der Zwischenzeit unsichtbar schreiben. Das ist genau das, was wir brauchen. Einen Haken hat die Sache aber noch. Man kann dann nicht mehr den einfachen Löschbefehl des GDPs anwenden, denn der löscht nur sichtbare Seiten. Wenn man eine unsichtbare Seite löschen will, so gibt es zwei Möglichkeiten. Entweder verwendet man das Unterprogramm CLPG, das aber etwa $\frac{1}{10}$ s für den Löschvorgang benötigt. Man kann aber auch einfach das alte Bild einer Seite dieser nochmals einschreiben, aber mit ERAPEN, und es dadurch löschen. Das geht sehr schnell und so ist es im Programm nach *Bild 11* auch realisiert.

Mit dem Unterprogramm NEWPAGE kann man die Bildseiten auswählen, im Register D0 steht die Schreibseite (0...3) und im Register D1 die Leseseite (0...3). Mit ERAPEN kann der Schreibstift des GDP auf Löschen geschaltet werden und mit SETPEN auf Schreiben.

Eine Besonderheit beim Löschen einer Bildseite ist, daß man den Würfel der vorhergehenden Position löschen muß, diese liegt aber zwei Drehungen zurück, da die letzte Drehung auf der gerade sichtbaren Bildseite eingestellt war. *Bild 12* zeigt das Ergebnis, nachdem alle Winkel durchlaufen sind.

Sehr schön wäre jetzt noch eine Steuerung mit einem Tastenfeld. *Bild 13* zeigt ein Programm und *Bild 14* einen Bildausschnitt. Das Tastenfeld wird an Port 30h der IOE-Baugruppe angeschlossen, wie bei der Mondlandung. Man kann jetzt neben dem Winkel auch den Augabstand oder die Perspektive einstellen. Taste 0 und 1 stellen den Winkel ein, die Tasten 2 und 3 den Augabstand und mit 4 und 5 kann man die Perspektive verändern.

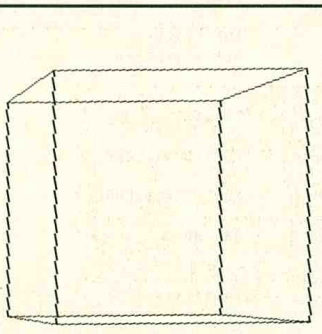


Bild 14. Beim Programmlauf

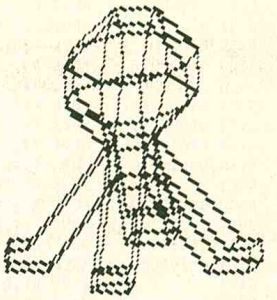


Bild 15. So sieht die Fähre aus

Bild 15 zeigt noch eine sehr komplizierte Landefähre aus unserer Werkstatt. Wenn man umfangreiche Grafiken darstellen will, so muß man jede Koordinate jeder Kante in eine Tabelle eingeben

Aufgaben

1. Konstruieren Sie ein Programm, bei dem nicht immer zwei Eckpunkte einer Kante eingegeben werden müssen, sondern die Eckpunkte von Polygonen angegeben werden können. Hinweise:
Dabei kann man zwei Tabellen verwenden. In einer Tabelle stehen nur die Eckpunkte und in einer anderen Tabelle steht die Nummer des Eckpunktes. Dadurch kann man die Ausgabegeschwindigkeit praktisch verdoppeln. Denn bei unserem Würfel wurden die Formeln auf jeden Eckpunkt zweimal angewendet, da jeder Eckpunkt von zwei Kanten berührt wird. Man muß bei dem neuen Verfahren aber wie im Struktogramm nach Bild 7 verfahren, denn man muß zunächst alle Eckpunkte berechnen, diese dabei abspeichern und kann dann erst in der Ausgaberoutine die Verbindungslinien anhand der Polygonliste bestimmen.

2. Konstruieren Sie einen einfachen Flugsimulator, indem Sie zunächst eine kleine Landschaft mit Flughafen definieren. Dann kann man das Flugzeug über die Landschaft bewegen, indem man zum Beispiel die Landschaft verschiebt. Bewegungen des Flugzeuges werden in Transformationen der Landschaft umgesetzt.
Außer drei Drehformeln, für die x-, y- und z-Achse benötigt man dazu nur eine Verschiebeformel, die vor der Drehung die Landschaft verschiebt und nachher zurücksetzt. Die Aufgabe ist nicht ganz einfach und man sollte sie in ganz kleinen Schritten angeben. Man braucht sicher auch ein Geometriebuch dazu, um sich nochmals mit der Vektorrechnung vertraut zu machen.
Am Schluß des 68008-Assemblerteils sei gesagt, daß es sicher noch viel zu berichten gäbe. Man sollte sich daher mit den Assemblerhandbüchern des 68008 (68000) auseinandersetzen und die Befehlsliste des Grundprogramms genau studieren. Die Befehlsliste ist im Anhang in Kurzform abgedruckt. In ausführlicher Form erhält man sie automatisch, wenn man das Grundprogramm in EPROMs bekommt (ab Version 4.3).

Rolf-Dieter Klein

Krater in Pascal

Anfang der 70er Jahre wurde von Prof. Niklaus Wirth an der ETH Zürich die Programmiersprache Pascal aus der Taufe gehoben. Weshalb man höhere Programmiersprachen benötigt und welchen Vorteil sie haben, das zu verstehen ist Ziel der nächsten Folgen.

Bisher haben Sie in Assembler programmiert. Damit kann man sehr schnelle Programme schreiben, doch hat das auch seine Haken. Große Programme können leicht unübersichtlich werden und man kann die Programme nicht auf andere Rechner übertragen, wenn diese nicht den gleichen Assembler verstehen.

Eine höhere Programmiersprache ist nun unabhängig vom unterliegenden Prozessor und kann damit auf unterschiedlichen Rechnern laufen. Außerdem gibt es in höheren Programmiersprachen Anweisungen, wie IF THEN ELSE oder REPEAT UNTIL, mit denen sich Struktogramme leicht als Programm verwirklichen lassen. Für alle Struktogrammformen, die wir bisher kennengelernt haben, entfällt dann die mühevolle Umsetzung in Assemblercode.

Eine weitere Aufgabe einer höheren Programmiersprache ist die Bereitstellung von Datenstrukturen. Felder und mehrdimensionale Felder zum Beispiel, die man in Assembler nur mit Mühe realisieren kann. Modernere Programmiersprachen wie Pascal können auch mathematischen Beweisführungen entgegenkommen, denn eine schwierige Aufgabe ist es, nachzuweisen, ob ein Problem korrekt in ein Programm umgesetzt wurde. Das kann man nur, wenn die Programmiersprache so genauen Regeln gehorcht, daß man die Korrektheit eines Programmes weitgehend formal beweisen kann.

Tabelle 1. Vergleich Assembler und Compiler

Assembler	Compiler A
MOVE D0,D1	A1:=B+5
↓	↓

3200 Maschinencode	MOVE B,D0
	ADD #5,D0
	MOVE D0,A1

	↓
	Maschinencode

Von Computern und Interpretern

Um eine Programmiersprache auf einem Rechner benutzbar zu machen, benötigt man immer ein Hilfsprogramm. Es gibt zwei Arten solcher Hilfsprogramme, einmal die Compiler und einmal die Interpreter. *Tabelle 1* zeigt ein Schema. Es zeigt, daß ein Assembler ebenfalls schon ein einfaches Übersetzungsprogramm war.

Ein Assembler übersetzt die Mnemonics einer Eingabezeile in den entsprechenden Maschinencode. Dabei ist dieser Übersetzungsvorgang praktisch nur eine Zuordnungsaufgabe. Einem Text wird ein bestimmter Code zugeordnet. Ein Compiler, der eine höhere Programmiersprache übersetzen soll, hat sehr viel mehr zu tun. Da gebe es zum Beispiel die Zuweisung A1:=B+5. Einer Variablen A1 soll die Summe der Variablen B und des Wertes 5 zugewiesen werden. Der Compiler muß diese Anweisung zunächst in Assemblercode übersetzen. Daraus ergeben sich die Anweisungen, die in *Tabelle 1* gezeigt sind. Danach muß der Assembler gestartet werden, der diese Anweisungen in den Maschinencode übersetzt. Es gibt auch Compiler, die den Assembler gleich beinhalten und nicht den Umweg über einen lesbaren Assemblertext gehen, sondern gleich Maschinencode erzeugen.

Bei Interpretern wiederum wird eine Anweisung vom Interpreter gelesen und gleich als Befehl ausgeführt. Eine Addition zum Beispiel wird also gleich durchgeführt und das Ergebnis abgespeichert. Dann holt sich der Interpreter den nächsten Befehl usw. Ein Compiler übersetzt ein Programm zunächst vollständig bevor es gestartet wird, ein Interpreter übersetzt immer nur ein kleines Stückchen, führt dieses dann direkt aus, ohne Maschinencode dafür zu erzeugen.

Dann übersetzt er das nächste Stückchen und so weiter. Das hat den Nachteil, daß die Ausführung eines Programms länger dauert als bei einem Compiler, denn wenn man z. B. eine Schleife hat, die 100 Additionen ausführt, so muß die Addition beim Interpreter 100mal übersetzt werden, während ein Compiler die Anweisung nur einmal übersetzt. Der Übersetzungsvorgang dauert aber meist länger als die eigentliche Ausführungszeit in Maschinensprache.

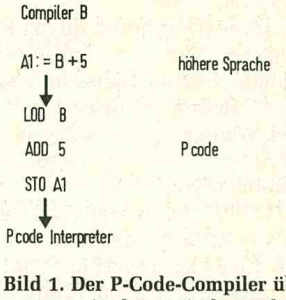


Bild 1. Der P-Code-Compiler übersetzt nur in den Zwischencode

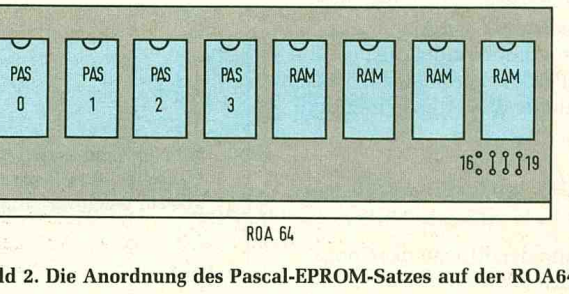


Bild 2. Die Anordnung des Pascal-EPROM-Satzes auf der ROA64

Zwischencode

Es gibt noch eine Mischung aus beiden Verfahren, den sogenannten Zwischencode-Compiler. Bei Pascal spricht man vom P-Code-Compiler. Dabei wird ein Programm wie beim Compiler zunächst vollständig übersetzt. Es wird aber nicht der Maschinencode für den Mikroprozessor erzeugt, sondern ein besonderer Code, der besonders leicht zu erzeugen ist. Wenn dieser Code dann erzeugt ist *Bild 1*, wird er von einem Interpreter interpretiert, der sich nun Befehl für Befehl holt und ausführt. Diese Mischform ist wesentlich schneller als ein reiner Interpreter, aber langsamer als ein Compiler, der direkt Maschinencode für den Prozessor erzeugt (auch Native Code Compiler genannt). Ein P-Code-Compiler hat den Vorteil, daß man ihn leicht auf andere Computer übertragen kann, da man nur den Interpreter neu schreiben muß, die Erzeugung des P-Code aber gleichbleibt. Daher hat sich die Sprache Pascal leicht verbreiten können, denn dafür gibt es zahlreiche P-Code-Compiler.

Pascal für den NDR-Computer

Auch wir haben einen solchen P-Code-Compiler für den NDR-Klein-Computer in EPROMs zur Verfügung. Zum Betrieb benötigt man zusätzlich 32 KByte RAM-Speicher, der zusammen mit dem EPROM-Satz auf einer ROA64-Baugruppe Platz findet.
Bild 2 zeigt die Anordnung auf der ROA-Baugruppe. Die Karte wird mit Adresse $10000 angesprochen. Daher sind die Brücken 17, 18 und 19 eingesetzt, 16 bleibt frei. Wenn man aber schon einen großen RAM-Speicher auf diesem Adreßbereich besitzt (z. B. von $8000...$1FFFF), dann kann man sich den Kauf der RAM-Bausteine sparen und stellt die Baugruppe einfach auf eine freie Adresse, z. B. $90000, ein. Der Pascal-Compiler ist relokativ, das heißt, der EPROM-Satz kann irgendwo im Hauptspeicher stehen. Nur das RAM bei Adresse $18000...$1FFFF muß vorhanden sein. Diese Adresse liegt fest und kann nicht geändert werden.
Das Grundprogramm findet die Position der Pascal-EPROMs automatisch, so daß man sonst nichts weiter beachten muß.

Der Compiler wird in Betrieb genommen

Testen wir nun den Einbau des Compilers. Dazu wird die Bibliotheksfunktion aufgerufen. Wenn *Bild 3* erscheint, wurde der Pascal-Compiler gefunden. Die Startadresse gibt die absolute Position des Compilers aus (wohin man den EPROM-Satz gesteckt hat). Jetzt kann man den Compiler starten, indem man J eingibt. Er erscheint (*Bild 4*) mit einer Fehlermeldung. Der Compiler sagt: „program incomplete". Das bedeutet „Programm unvollständig". Der Compiler will nämlich immer ein mit dem Editor eingegebenes Programm hernehmen und übersetzen. Findet er keines, so beschwert er sich mit der Meldung oben.

```
Bibliothek

Name   : PASCAL/S
Start  : 0900F8
Laenge : 008000

Starten J = JA
cr = weiter, M = Menue
```

Bild 3. Die Bibliothek enthält jetzt Pascal

Bevor wir uns an das Programmieren in Pascal machen, geben wir ein kleines Testprogramm ein, um die Funktion des Compilers endgültig zu prüfen. *Bild 5* zeigt das Programm, so wie man es im Editor eingibt. Achtung, keine Punkte, Strichpunkte oder Kommas vergessen, der Compiler beschwert sich sonst bei der Übersetzung! Jetzt wird der Compiler über die Bibliotheksfunktion gestartet. Es ergibt sich *Bild 6* als Ausgabe, wenn man die Zahl 0.3141 eingibt, sobald der Cursor unter dem Text „Start P-Code-Interpreter 3.1" blinkt. Die Eingabe wird wie gewöhnlich mit einem CR (carriage return) abgeschlossen.
Hier noch ein allgemeiner Hinweis. Es ist unmöglich, hier alle Pascal-Besonderheiten und Befehle genau zu behandeln, da die Sprache sehr umfangreich ist und es viele Einzelheiten gibt. Daher sei hier auf die Pascal-Fachliteratur verwiesen.

Mondfähre in Pascal

Aufgabe soll es sein, eine Mondfähre zu landen. Dabei soll die Ausgabe in Form von Zahlenwerten auf dem Bildschirm erscheinen. Als Eingabe wird der Schub eingegeben. *Tabelle 2* zeigt das Pflichtenheft.

```
**********************************
* PASCAL/S  Pcode-Compiler V3.1 *
* (C) 1984 Rolf-Dieter Klein    *
* Version nach PASCAL/S von     *
* N.Wirth 1976, E.T.H Zuerich   *
**********************************

   0
 program incomplete

 key words
```

Bild 4. Wenn man den Compiler ohne Programm startet, beschwert er sich

Tabelle 2. Das Pflichtenheft zur Mondlandung

1. Anzeige: Höhe
2. Anzeige: Treibstoffmenge
3. Anzeige: Sinkgeschwindigkeit

In *Bild 7* ist das Pascal-Programm dargestellt, das die Aufgabe erfüllt und *Bild 8* zeigt ein paar Eingaben, bei der die Fähre auch noch weiter steigt.
Jedes Pascal-Programm muß mit dem Text „PROGRAM" beginnen. Daran erkennt der Compiler den Anfang des Programms. Danach muß der Name des Programms stehen. Dann folgen in Klammern zwei Angaben: INPUT sagt dem Compiler, daß in dem Programm Eingaben von der Tastatur vorkommen, und OUTPUT sagt dem Compiler, daß auch Ausgaben auf dem Bildschirm gemacht werden sollen.

Danach kommt ein Strichpunkt. Er wird immer zur Trennung von Befehlen verwendet, denn in Pascal dürfen auch mehrere Befehle in einer Zeile stehen. Jetzt muß der sogenannte Vereinbarungsteil folgen, der in Pascal immer direkt nach PROGRAM folgen muß. Mit CONST wird die Konstantenvereinbarung eingeleitet. Sie entspricht im Assembler der EQU-Anweisung. Man kann damit Namen anstelle von Zahlenwerten verwenden.

Rechnen mit Gleitkommazahlen

Dann kommt der Variablendeklarationsteil, durch den Speicherplatz für Variable reserviert wird. Die Variable HOEHE bekommt einen Speicherplatz vom Typ REAL. Das bedeutet, in der Variablen

```
program kleinesprog(input,output);
 var a:real;
begin
 readln(a);
 writeln(a,' ',sin(a),' ',sqrt(a));
end.
```

```
Textstart=009000 Fenster=009000 Tor=009000 einf amer CTRL-J=Hilfe
```

Bild 5. Ein kleines Testprogramm

```
*********************************
* PASCAL/S  Pcode-Compiler V3.1 *
* (C) 1984 Rolf-Dieter Klein    *
* Version nach PASCAL/S von     *
* N.Wirth 1976, E.T.H Zuerich   *
*********************************

    0  program kleinesprog(input,output);
    0   var a:real;
    0  begin
    0   readln(a);
    3   writeln(a,' ',sin(a),' ',sqrt(a));
   16  end.



Start PCODE-Interpreter 3.1
0.3141
    3.141000000000e-01      3.089606270534e-01      5.604462507680e-01
```

Bild 6. Das Ergebnis nach der Übersetzung (oben) und nach einem Programmlauf (unten) von Kleinesprog

```
*********************************
* PASCAL/S  Pcode-Compiler V3.1 *
* (C) 1984 Rolf-Dieter Klein    *
* Version nach PASCAL/S von     *
* N.Wirth 1976, E.T.H Zuerich   *
*********************************

    0  program mondlandung(input,output);
    0
    0  const
    0   grav=1.6;
    0   anfhoehe=1000;
    0   anftreib=80;
    0
    0  var
    0   hoehe   : real;
    0   treibst : real;
    0   geschw  : real;
    0   schub   : real;
    0
    0  begin
    0   writeln('Mondlandespiel');
    3   hoehe :=  anfhoehe;
    7   geschw  :=  0;
   11   treibst := anftreib;
   15   repeat
   15    if treibst <0 then treibst := 0;
   24    writeln('Hoehe:       ',hoehe:8:2,' m');
   33    writeln('Geschw.:     ',geschw:8:2,' m/s');
   42    writeln('Treibstoff: ',treibst:8:2);
   49    writeln;
   50    repeat
   50     write('Schub in m/s (-9..+9): ');
   52     read(schub);
   54     writeln;
   55    until abs(schub)<10;
   61    if treibst > 0
   63     then
   66       geschw:=geschw+schub;
   71    geschw:=geschw-grav;
   76    hoehe:=hoehe+geschw;
   81    treibst:=treibst-1-abs(schub)*0.1;
   92   until hoehe <=  0;
   97   if (abs(hoehe)<0.5) and (abs(geschw)<1)
  106    then
  108     writeln('Gut gelandet')
  110    else
  111     begin writeln('Bruchlandung mit');
  115       writeln(geschw:5:2,' m/s');
  122       writeln(hoehe:5:2,' m');
  129     end;
  129  end.
```

Bild 7. Das Mondlandespiel (nur Text zwischen „PROGRAM" und „END." eingeben, nicht die Zahlen links)

```
Start PCODE-Interpreter 3.1
Mondlandespiel
Hoehe:        1000.00 m
Geschw.:         0.00 m/s
Treibstoff:     80.00

Schub in m/s (-9..+9): 3

Hoehe:        1001.40 m
Geschw.:         1.40 m/s
Treibstoff:     78.70

Schub in m/s (-9..+9): 7

Hoehe:        1008.20 m
Geschw.:         6.80 m/s
Treibstoff:     77.00

Schub in m/s (-9..+9):
```

Bild 8. Start des Mondlandeprogramms

HOEHE kann man eine Gleitkommazahl, also eine Zahl, die auch Nachkommastellen und einen Exponenten besitzen kann, speichern. Das ist nun neu gegenüber unserem Assembler. Der 68008 kann von sich aus nicht mit Gleitkommazahlen umgehen. Dazu befinden sich im Pcode-Interpreter umfangreiche Programmteile, die das verwirklichen, denn in der Sprache Pascal kann man sehr wohl REAL-Zahlen verwenden. Es gibt in Pascal aber auch INTEGER-Zahlen, die einer 16-Bit-Variablen im Assembler entsprechen. Die Gleitkommazahlen haben bei uns 12 Stellen (mit Schutzstellen 14) und einen Exponenten, der maximal 126 und minimal –126 betragen kann.

Begin und End

Nach dem Deklarationsteil folgt das Hauptprogramm. Das Hauptprogramm fängt immer mit dem Schlüsselwort „BEGIN“ an und hört mit „END.“ auf. Der Punkt nach END ist wichtig und darf nicht vergessen werden, sonst erscheint die Meldung „program incomplete“. Das Programm beginnt mit einer WRITELN-Anweisung. WRITELN heißt WRITE WITH LINEFEED und bedeutet, es soll eine Ausgabe gemacht werden, und der Cursor anschließend in die nachfolgende Zeile gesetzt werden. In Klammern steht danach das Auszugebende. Texte müssen dabei zusätzlich in Hochkommas eingeschlossen werden. Mit dem Zeichen „:=“ erfolgt eine Zuweisung eines arithmetischen Ausdrucks an eine Variable, ähnlich wie im Grundprogramm. Dann folgt die Anweisung REPEAT. Dazu gehört die Anweisung UNTIL, die bei der P-Code-Adresse 92 steht. Die Zahlen auf der linken Seite sind die Anzahl der P-Code-Befehle, die bisher erzeugt wurden.
REPEAT UNTIL realisiert eine Wiederholschleife, die dem „Wiederhole bis“ exakt entspricht. Der Spielblock soll nämlich solange wiederholt werden, bis

```
                    Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

009C00
009C00                      ;
009C00                      ; ASSEMBLER HILFSROUTINEN FUER PASCAL-MONDLANDUNG
009C00                      ;
009C00
009C00                      TAB1:
009C00  08 02 09 00 00      DC.B 8,2,9,0,0,7,3,2,2,4,4,4,4,6,6,5,1,0,0,10
009C05  07 03 02 02 04
009C0A  04 04 04 06 06
009C0F  05 01 00 00 0A
009C14                      TAB2:
009C14  02 09 00 00 07      DC.B 2,9,0,0,7,3,2,2,4,4,4,4,6,6,5,1,0,0,10
009C19  03 02 02 04 04
009C1E  04 04 06 06 05
009C23  01 00 00 0A
009C27
009C27
009C27
009C27  00                  DS.B 0
009C28
009C28
009C28  00009C14            TABVAR: DC.L TAB2
009C2C
009C2C                      ASSFAEHRE:
009C2C  2079 00009C28        MOVEA.L TABVAR,A0
009C32  3401                 MOVE D1,D2
009C34  3200                 MOVE D0,D1
009C36  303C 0003            MOVE #$3,D0
009C3A  4EB9 0000326C        JSR @FIGUR
009C40  4E75                 RTS
009C42
009C42                      ASSAUS:
009C42  23FC 00009C00        MOVE.L #TAB1,TABVAR
009C48  00009C28
009C4C  4E75                 RTS
009C4E
009C4E
009C4E                      ASSAN:
009C4E  23FC 00009C14        MOVE.L #TAB2,TABVAR
009C54  00009C28
009C58  4E75                 RTS
009C5A
009C5A
009C5A                      TABMOND:
009C5A  0002 0005 0000       DC.W  2,5,0,0,7,2,5,4,3,9,0,0,0,3,0,0,0,0,0
009C60  0000 0007 0002
009C66  0005 0004 0003
009C6C  0009 0000 0000
009C72  0000 0003 0000
009C78  0000 0000 0000
009C7E  0000
009C80
009C80                      ASSMOND:
009C80  303C 0001            MOVE #1,D0
009C84  4241                 CLR D1
009C86  4EB9 00000E0C        JSR @NEWPAGE
009C8C  4EB9 00009CA6        JSR ASS1
009C92  303C 0000            MOVE #0,D0
009C96  4241                 CLR D1
009C98  4EB9 00000E0C        JSR @NEWPAGE
009C9E  4EB9 00009CA6        JSR ASS1
009CA4  4E75                 RTS
009CA6
009CA6
009CA6
009CA6                      ASS1:
009CA6  323C 0190            MOVE #400,D1
009CAA  4242                 CLR D2
009CAC  4EB9 00000EE2        JSR @MOVETO
009CB2  41F9 00009C5A        LEA TABMOND,A0
009CB8                      AK1:
009CB8  3418                 MOVE (A0)+,D2
009CBA  4EB9 00000F4C        JSR @DRAWTO
009CC0  0641 000A            ADD #10,D1
009CC4  0C41 01FE            CMP #510,D1
009CC8  6600 FFEE            BNE AK1
009CCC  4E75                 RTS
009CCE
009CCE                      ASSKRATER:
009CCE  303C 000C            MOVE #12,D0
009CD2  4EB9 00000D66        JSR @CMD
009CD8                      WA1:
009CD8  4EB9 00000E38        JSR @SYNC
009CDE  6700 FFF8            BEQ WA1
009CE2                      WA2:
009CE2  4EB9 00000E38        JSR @SYNC
009CE8  6700 FFF8            BEQ WA2
009CEC  4EB9 00004020        JSR @CLRSCREEN
009CF2  4E75                 RTS
009CF4
009CF4

008B38  Ende-Symboltabelle
```

Bild 9. Assemblerunterprogramme für das erweiterte Spiel

die Höhe der Mondlandefähre kleiner oder gleich 0 ist. Mit der Anweisung IF kann man eine Verzweigung realisieren.

Wenn noch ein ELSE folgt, so bedeutet das, daß der ELSE-Teil durchlaufen wird, wenn die Bedingung hinter IF nicht erfüllt war. Im Programm gibt es auch noch eine Eingabeanweisung, nämlich READ, mit der der Wert der Variablen SCHUB eingelesen wird. Bei der Ausgabe von Werten mit WRITELN ist eine Ausgabeformatierung möglich. Bei in Zeile 122 z. B. wird die Höhe mit insgesamt 5 Stellen und 2 Nachkommastellen ausgeben. Die Zahl hinter dem Doppelpunkt nach der Variablen gibt die Gesamtzahl der Stellen an und die nächste Zahl die Zahl der Nachkommastellen. Läßt man sie beide weg, so wird die Zahl im Exponentialformat ausgegeben. In Zeile 97 gibt es noch eine Funktion, die Ihnen nicht bekannt ist. Mit ABS ( ) kann man den Absolutbetrag bilden.

Etwas Grafik

Nun wäre es ganz schön, wenn man auch noch eine grafische Ausgabe auf den Bildschirm bringen könnte. Die Sprache Pascal bietet dazu leider keine direkten Möglichkeiten, doch gibt es eine Besonderheit in unserem Grundprogramm, die Grafik möglich macht. Wenn man zum Beispiel ein Zeichen mit dem Code 1 auf den Bildschirm ausgeben will (dies ist ein nicht druckendes ASCII-Steuerzeichen), so leitet das Grundprogramm alle weiteren Ausgaben in einen kleinen Zwischenspeicher. Erst wenn ein Zeilenvorschub, z. B. am Ende von WRITELN ausgegeben wird, geschieht etwas. Dann wird nämlich der Inhalt des Puffers analysiert. Das zweite Zeichen wird dabei vom Grundprogramm als Befehl aufgefaßt. Wenn dort ein „E" steht, so bedeutet das Execute, also ausführen, und zwar von dem Assemblerprogramm, dessen Name jetzt nach dem E folgt. Wenn man an dieses Unterprogramm noch Werte übergeben will, so muß man sie dahinter schreiben, jeweils durch ein Leerzeichen getrennt.

Man kann zum Beispiel

WRITELN(chr(1),'E @SCHREITE 100');

im Pascal-Programm eingeben. Es wird dann eine senkrechte Linie gezeichnet. Mit chr( ) wird der Code des in Klammern stehenden Wertes an das Grundprogramm ausgegeben. 1 ist der Code für die Spezialoperation.

Die Parameter werden nacheinander in die Register D0.L...D7.L übertragen, je nachdem, wieviele Parameter angegeben wurden. Damit gibt es also einen einfachen Weg Assemblerunterprogramme von Pascal aus aufzurufen, wie auch direkt Unterprogramme aus dem Grundprogramm zu verwenden.

```
**********************************
* PASCAL/S  Pcode-Compiler V3.1 *
* (C) 1984 Rolf-Dieter Klein    *
* Version nach PASCAL/S von     *
* N.Wirth 1976, E.T.H Zuerich   *
**********************************

    0  program mondlandung(input,output);
    0
    0  const
    0   grav=1.6;
    0   anfhoehe=1000;
    0   anftreib=80;
    0
    0  var
    0   x,y     : integer;
    0   hoehe   : real;
    0   treibst : real;
    0   geschw  : real;
    0   schub   : real;
    0
    0  procedure faehre(x,y:integer);
    0  begin
    0   writeln(chr(1),'E assfaehre ',x,' ',y)
   11  end;
   12
   12  procedure krater;
   13  begin
   13   writeln(chr(1),'E asskrater')
   18  end;
   19
   19  procedure mond;
   20  begin
   20   writeln(chr(1),'E assmond')
   25  end;
   26
   26  procedure triebaus;
   27  begin
   27   writeln(chr(1),'E assaus');
   33  end;
   33
   33  procedure trieban;
   34  begin
   34   writeln(chr(1),'E assan')
   39  end;
   40
   40
   40  begin
   41   writeln(chr(1),'E @clrscreen');
   47   writeln('Mondlandespiel');
   50   mond;
   52   trieban;
   54   hoehe :=  anfhoehe;
   58   geschw  :=  0;
   62   treibst := anftreib;
   66   repeat
   66    if treibst<0 then treibst:=0;
   75    x:=480;
   78    y:=trunc(hoehe/4);
   85    faehre(x,y);
   89    writeln('Hoehe:       ',hoehe:8:2,' m');
   98    writeln('Geschw.:     ',geschw:8:2,' m/s');
  107    writeln('Treibstoff: ',treibst:8:2);
  114    writeln;
  115    repeat
  115     write('Schub in m/s (-9..+9): ');
  117     read(schub);
  119     writeln;
  120    until abs(schub)<10;
  126    if treibst > 0
  128     then
  131       geschw:=geschw+schub
  133     else
  136       triebaus;
  139    geschw:=geschw-grav;
  144    hoehe:=hoehe+geschw;
  149    treibst:=treibst-1-abs(schub)*0.1;
  160   until hoehe <=  0;
  165   if (abs(hoehe)<0.5) and (abs(geschw)<1)
  174    then
  176     writeln('Gut gelandet')
  178    else
  179     begin
  180      krater;
  182       writeln('Bruchlandung mit');
  185       writeln(geschw:5:2,' m/s');
  192       writeln(hoehe:5:2,' m');
  199     end;
  199  end.
```

Bild 10. Pascal-Programm für Grafik mit Assemblerunterprogrammen

```
Hoehe:          955.20 m
Geschw.:        -11.20 m/s
Treibstoff:      73.00

Schub in m/s (-9..+9): 5

Hoehe:          947.40 m
Geschw.:         -7.80 m/s
Treibstoff:      71.50

Schub in m/s (-9..+9): 0

Hoehe:          938.00 m
Geschw.:         -9.40 m/s
Treibstoff:      70.50

Schub in m/s (-9..+9): 0

Hoehe:          927.00 m
Geschw.:        -11.00 m/s
Treibstoff:      69.50

Schub in m/s (-9..+9): 2
```

Bild 11. Das Ergebnis auf dem Bildschirm

Maschinenprogramme einbinden

Bild 9 zeigt die Assemblerliste der Unterprogramme, die für das Mondlandespiel gebraucht werden. Mit Hilfe des FIGUR-Befehl wird die Mondlandefähre dargestellt. *Bild 10* zeigt das Pascal-Programm mit den Aufrufen. Zum Start gibt man zuerst das Assemblerprogramm ein und übersetzt es. Dann gibt man das Pascal-Programm auf eine neue Editor-Adresse (Textstart wählen) oder man überschreibt das alte Programm und startet den Pascal-Compiler. Danach zeigt sich z. B. *Bild 11*. Wenn Namen, die in Pascal aufgerufen werden, nicht definiert sind (wenn man sie falsch geschrieben hatte oder weil sie vergessen wurden), so werden die Unterprogramme einfach nicht ausgeführt, der Pascal-Lauf wird allerdings nicht abgebrochen. Damit kann man jedoch zunächst einmal das Pascal-Programm entwickeln und erst später die Assemblerteile hinzufügen.

Übrigens kann man das Pascal-Programm auch testen, indem man zunächst den Befehl CHR(1) wegläßt, dann werden Befehl, Programmname und Parameter auf dem Bildschirm ausgegeben und man kann kontrollieren, ob Leerzeichen stimmen usw.
In unserem Programm nach Bild 10 gibt es noch etwas Neues. In Pascal kann man Unterprogramme mit dem Schlüsselwort PROCEDURE definieren. Dies geschieht nach dem Vereinbarungsblock des Hauptprogramms und noch vor dem Hauptprogramm. Unterprogramme benötigen meist auch Parameter. Die Namen und der Typ dieser Parameter werden in Klammern angegeben. FAEHRE zum Beispiel hat zwei Parameter, x und y, vom Typ REAL. Die Parameter werden hier an das Assemblerunterprogramm ASSFAEHRE weitergereicht und kommen dort im Register D0.L und D1.L an. Man kann hier nur Integerzahlen durchreichen, da im Assembler keine Gleitkommazahlen existieren.

Aufgaben:

1. Spielen Sie einmal Compiler und übersetzen Sie die folgenden Befehle einer Sprache, die auch für den NDR-Computer vorhanden ist (GOSI), in Assemblercode:

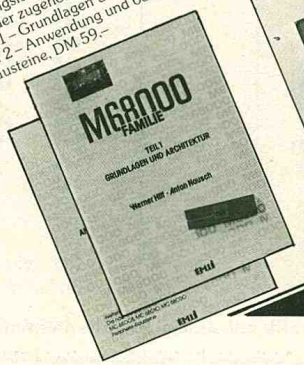
WIEDERHOLE 4 [VORWAERTS 100 RECHTS 90]
Dabei sollen Elemente der Zeichensprache verwendet werden.

2. Wie kann man Parameterübergaben in Assembler realisieren?

te-wi aktuell...

UNIX™-ANWENDERHANDBUCH
(Thomas/Yates)
Das Werk ist so praxisnah erstellt, daß der Leser bereits nach kurzer Zeit die Arbeit mit seinem Computer aufnehmen kann. Aber auch ohne eine Anlage wird durch zahlreiche Bildschirmbeispiele ein Wissen vermittelt, das Einblick in die Arbeitsweise von UNIX verschafft.
UNIX-Anwenderhandbuch, System III-kompatibel. 478 Seiten, Softcover. DM 79.–

M68000 FAMILIE
(W.Hilf, A.Nausch)
Dieses zweibändige Werk befaßt sich grundlegend mit der M68000-Familie, einer Familie von leistungsfähigen 16/32 Bit-Prozessoren und der zugehörigen Peripherie.
Teil 1 – Grundlagen und Architektur. DM 79.–
Teil 2 – Anwendung und 68000 Bausteine. DM 59.–



Die C-SPRACHE
(Stefan Lösch/Zdenek Stanka)
Bei äußerst geringem Wortschatz verleiht die ungewöhnliche Logik der Sprache „C" die unbegrenzten Kombinationsmöglichkeiten. Das Buch „Die C-Sprache" dient der praktischen Konzipierung von „C"-Programmen. Logik und Struktur dieser Sprache stehen im Mittelpunkt.
300 Seiten, Softcover. DM 59.–

tm 4154

te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40
te-wi

Rolf-Dieter Klein

Die Türme von Hanoi

In dieser Folge geht es um etwas Mathematik. Daraus wird sich die Technik der Rekursion ergeben, die häufig von modernen Computerprogrammen benutzt wird. Diese Technik erlaubt es, in wenigen Programmzeilen Abläufe zu programmieren, die sehr komplexe Tätigkeiten darstellen.

Es sei mit einer mathematischen Definition begonnen: Man nehme die natürlichen Zahlen einschließlich 0, also 0, 1, 2, 3, 4... Diejenige Funktion F, die jeder natürlichen Zahl n, sofern sie verschieden von 0 ist, das Produkt $1 \cdot 2 \cdot 3 \cdot \ldots (n-1) \cdot n$ zuordnet und der 0 den Wert 1, heißt Fakultät.

Beispiele:
$F(2) = 1 \cdot 2 = 2$
$F(3) = 1 \cdot 2 \cdot 3 = 6$
$F(4) = 1 \cdot 2 \cdot 3 \cdot 4 = 24$
$F(0) = 1$

In den Mathematikbüchern wird oft anstelle von F einfach ein Ausrufezeichen hinter die Zahl, deren Fakultät gemeint ist, gesetzt: $F(4) = 4!$

Wenn Sie oben die ersten Werte genau anschauen, dann kann man ablesen, daß $F(4) = F(3) \cdot 4$, $F(3) = F(2) \cdot 3$ ist und so weiter. Mathematiker können die Pünktchen oben in der Definition nicht leiden, sie schreiben statt dessen: Die Funktion F mit $F(n) = F(n-1) \cdot n$ für alle $n \neq 0$ und mit $F(0) = 1$ heißt Fakultät. Oder: $n! = (n-1)! \cdot n$.

```
program fakultaet(input,output);

var i:integer;
    r:real;

function fac(a:real):real;
 begin
  if a=0
   then fac:=1.0
   else fac:=fac(a-1)*a
 end;

begin
 repeat
  writeln;
  write('fac von: ');
  read(i);
  writeln;
  r:=fac(i);
  writeln(' ist ',r);
  writeln(' ist ',r:14:0)
 until false
end.
```

Textstart=009000 Fenster=009000 Tor=009000 amer CTRL-J=Hilfe

Bild 1. Das Programm Fakultät als rekursives Programm

```
Start PCODE-Interpreter 3.1

fac von: 3

 ist      6.000000000000e+00
 ist               6.

fac von: 10

 ist      3.628800000000e+06
 ist        3628800.
```

Bild 2. Beispiele (maximale Eingabe: 84)

Bild 1 zeigt ein Pascal-Programm, das für jede vernünftige Eingabe i den Wert i! ausrechnet. Vernünftig sind die Eingaben, bei welchen i nicht zu groß wird. *Bild 2* zeigt einen Probelauf.

Im Programm wurde mit FUNCTION die Funktion fac definiert. Als Parameter wird die Variable a in Klammern angegeben. Nun steht im Programmteil von fac einfach die Formulierung, wie sie in der Definition steht. Wenn a=0 ist, also 0! berechnet werden soll, so erhält fac den Wert 1, das ist in diesem Fall der Wert, den die Funktion an das aufrufende Programm übergibt. War a nicht 0, so wird die rekursive Formel gebraucht, und fac ergibt sich aus fac von a minus 1 mal a. Man hätte auch a*(FAC(a–1)) schreiben können, die Wirkung ist die gleiche. Hier ruft sich die Funktion fac selbst auf, man sagt, fac ist rekursiv.

Damit bei einem neuen Durchlauf nicht der gleiche Speicherplatz für die Variable a verwendet wird, legt Pascal die Variablen auf einen Stack, und beim neuen Aufruf wird einfach ein neues Gebiet im Stack verwendet. Nur deshalb kann man eine Funktion überhaupt rekursiv verwenden.

Die Türme von Hanoi

Die Geschichte der Denksportaufgabe „Die Türme von Hanoi" ist sicher so

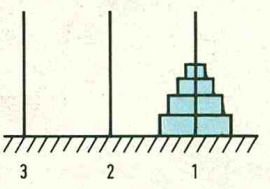


Bild 3. Die Türme von Hanoi, eine alte Aufgabe

```
*********************************
* PASCAL/S  Pcode-Compiler V3.1 *
* (C) 1984 Rolf-Dieter Klein    *
* Version nach PASCAL/S von     *
* N.Wirth 1976, E.T.H Zuerich   *
*********************************

    0  program hanoi(input,output);
    0   var total:integer;
    0
    0  procedure schiebe(hoehe,von,nach,mit : integer);
    0
    0  begin
    0   if hoehe > 0
    2    then begin
    4     schiebe(hoehe-1,von,mit,nach);
   13     writeln('von ',von,' nach ',nach);
   22     schiebe(hoehe-1,mit,nach,von);
   31    end;
   31   end;
   31
   31   begin
   32    repeat
   32     writeln;
   33     write('Turmhoehe ');
   35     read(total);
   37     writeln;
   38     schiebe(total,1,3,2);
   44    until false
   44  end.
```

Bild 4. Das Programm „Die Türme von Hanoi"

platz, auf dem zu diesem Zweck die Platten zwischengelagert werden können. Das ganze soll nämlich unter der Randbedingung stattfinden, daß während des Neuaufbaues nie eine größere Platte auf einer kleineren zu liegen kommt, auch nicht zeitweise, während des Zwischenlagerns.

Die Lösung der Aufgabe ist ebenfalls rekursiv. *Bild 4* zeigt das Programm. Das Programm ist erstaunlich kurz, dennoch nicht einfach zu verstehen. *Bild 5* zeigt einen Probeablauf mit 3 Scheiben. Bei der Lösung von rekursiven Problemen muß man möglichst immer zwei Dinge finden. Ein Abbruchkriterium, das ist hier gegeben, sobald die Turmhöhe kleiner oder gleich 0 ist, und die Rekursion für einen allgemeinen Fall.

Verfolgen Sie bitte mit Papier und Bleistift den Ablauf des Programms nach Bild 4, mit einer Turmhöhe von 2. Bei

bekannt, daß hier nicht wiederholt werden muß, daß die Mönche im Tempel in Hanoi dann den Weltuntergang herbeiführen, wenn sie ihre schwere Aufgabe gelöst haben, die mathematisch formuliert so lautet:

Gegeben ist (im Garten des Tempels) ein Turm aus aufeinanderliegenden Platten, von welchen die unterste die größte ist und wobei von je zwei aufeinanderliegenden die obere kleiner als die untere ist. Gegeben weiterhin neben dem Standplatz des Turmes ein weiterer Platz, auf dem er aus den Platten neu aufgebaut werden soll und ein Hilfs-

```
Start PCODE-Interpreter 3.1

Turmhoehe 3

von 1 nach 3
von 1 nach 2
von 3 nach 2
von 1 nach 3
von 2 nach 1
von 2 nach 3
von 1 nach 3

Turmhoehe
```

Bild 5. Der Ablauf mit Turmhöhe 3

```
                Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

009C00
00AB00                  ORG $AB00
00AB00
00AB00
00AB00                  ASSINIT:
00AB00  4EB9 00004020    JSR @CLRSCREEN
00AB06  4240             CLR D0
00AB08  4241             CLR D1
00AB0A  4EB9 00001EF4    JSR @SETFLIP
00AB10  4E75             RTS
00AB12
00AB12                  ASSPLATTE:    ; D0= HOEHE, D1=X , D2= Y
00AB12  41F9 0000AB28    LEA SCHEIBE,A0
00AB18  4EB9 0000326C    JSR @FIGUR
00AB1E  4E75            RTS
00AB20
00AB20                  ASSFEST:
00AB20  4EB9 00003266    JSR @SETFIG
00AB26  4E75             RTS
00AB28
00AB28
00AB28                  SCHEIBE:
00AB28  00 00 02 04 04   DC.B 0,0,2,4,4,4,4,6,0,0,10
00AB2D  04 04 06 00 00
00AB32  0A
00AB33
00AB33
00AB33
00AB33
00AB33
00AB33
00AB33
00AB33
00AB33
00AB33
00AB33
00AB33

008B80  Ende-Symboltabelle
```

Bild 6. Die Assemblerroutinen für eine grafische Ausgabe

```
*********************************
* PASCAL/S  Pcode-Compiler V3.1 *
* (C) 1984 Rolf-Dieter Klein    *
* Version nach PASCAL/S von     *
* N.Wirth 1976, E.T.H Zuerich   *
*********************************
    0  program hanoi(input,output);
    0
    0   const
    0    x1 = 200;  (* Lage der Tuerme in x-Richtung *)
    0
    0    y0 = 40;   (* Hoehe der Tuerme *)
    0    yoben = 100; (* Hoehe max der Tuerme *)
    0
    0
    0   var total:integer;
    0   x,y: integer;
    0
    0   h: array[1..3] of integer;  (* hoehe *)
    0
    0
    0  procedure setturm; (* Startturm *)
    0  var i:integer;
    0
    0  begin
    0   for i:=1 to total do begin
    4    writeln(chr(1),'E assplatte ',(1+total-i)*2,' ',x1,' ',y0+i*14);
   30    writeln(chr(1),'E assfest');
   36   end;
   37   writeln(chr(27),'=',chr(32+23),chr(32+31),' 1        2        3');
   55   h[1]:=total;
   60   h[2]:=0;
   65   h[3]:=0;
   70  end;
   70
   70  procedure platte(hoehe,von,nach:integer);
   71
   71  var x,y,i:integer;
   71   dx:integer;
   71   xziel:integer;
   71
   71  begin
   71   for y:=h[von]*14+y0 to yoben do
   82    writeln(chr(1),'E assplatte ',hoehe*3,' ',x1+50*(von-1),' ',y);
  107   h[von]:=h[von]-1;
  117   x:=x1+50*(von-1);
  126   xziel:=x1+50*(nach-1);
  135   if xziel < x then dx:=-1 else dx:=1;
  147   repeat
  147    writeln(chr(1),'E assplatte ',hoehe*3,' ',x,' ',yoben);
  165    x:=x+dx;
  170   until x=xziel;
  174   h[nach]:=h[nach]+1;
  184   for y:=yoben downto h[nach]*14+y0 do
  195    writeln(chr(1),'E assplatte ',hoehe*3,' ',x1+50*(nach-1),' ',y);
  220   writeln(chr(1),'E assfest');
  226  end;
  226
  226  procedure schiebe(hoehe,von,nach,mit:integer);
  227
  227  begin
  227   if hoehe > 0
  229    then begin
  231     schiebe(hoehe-1,von,mit,nach);
  240     writeln(chr(27),'=  ');
  246     writeln('von ',von,' nach ',nach);
  255     platte(hoehe,von,nach);
  261     schiebe(hoehe-1,mit,nach,von);
  270    end;
  270   end;
  270
  270   begin
  271    repeat
  271     writeln;
  272     write('Turmhoehe ');
  274     read(total);
  276     writeln(chr(1),'E assinit');
  282     setturm;
  284     writeln;
  285     schiebe(total,1,3,2);
  291    until false
  291  end.
```

Bild 7. Pascal-Programm „Die Türme von Hanoi" mit grafischer Ausgabe

```
von 1 nach 3

Turmhoehe 3
```

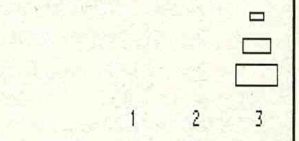


Bild 8. So sieht der Bildschirm aus

jedem erneuten Aufruf der Prozedur müssen alle Parameter der jeweiligen Prozedur gemerkt werden. (Die Lösung ist nicht einfach.)

Wenn man das Programm verstanden hat, kann man sich an eine komfortable Gestaltung der Ausgabe machen. *Bild 6* zeigt die Assembler-Unterprogramme für eine grafische Darstellung der Scheiben und *Bild 7* das dazugehörige Pascal-Programm. Die Scheiben werden dann nach dem Start auf dem Bildschirm so bewegt, wie es die Aufgabenstellung verlangt.

Bild 8 zeigt den Stand nach Ablauf, bei einer Turmhöhe von 3. Man kann bei dem Programm leider nicht beliebige Turmhöhen angeben, weil die Scheiben ja auf den Bildschirm passen müssen.

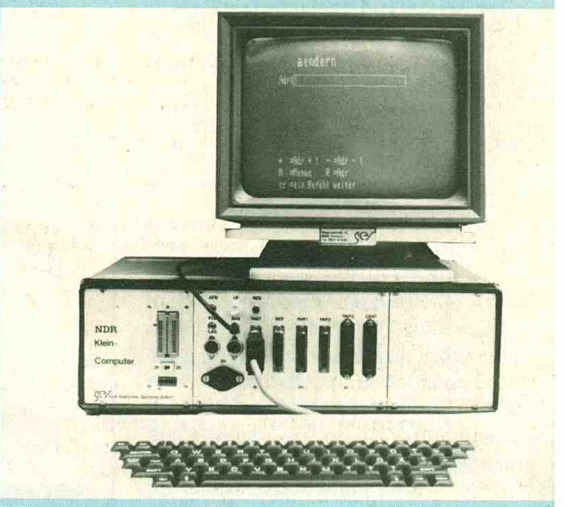
Aufgaben

1. Erweitern Sie das Programm, so daß auch die Dicke der Scheibe bei der Ausgabe berücksichtigt wird.

2. Erweitern Sie das Programm auf große Turmhöhen.

Den NDR-Computer erhalten

Wenn Sie wirklich wissen wollen, wie ein Computer funktioniert: Bauen Sie ihn doch einfach selbst!



Der NDR-Computer aus dem Fernsehen – ein Selbstbau-Computer mit unbegrenzten Möglichkeiten!

Steigen Sie klein ein mit dem NDR-Computer – schon für unter DM 400,– können Sie sich einen funktionsfähigen Computer selbst bauen, der später zum Profi-System in verschiedenen Variationen (z.B. 16 Bit oder CP/M) ausgebaut werden kann.

Das Einsteigerpaket: Bausätze DM 395,–

Lernen Sie mit dem NDR-Computer: Durch den Selbstbau lernen Sie wirklich, wie ein Computer funktioniert. Sie lernen auch bei der Programmierung: Beginnend bei der Maschinenprogrammierung im Einsteigerpaket bis zu allen wichtigen Programmiersprachen beim späteren Ausbau.

Natürlich gibt es auch alle Baugruppen fertig aufgebaut und geprüft.

Sie entscheiden sich für einen Computer, der nie veralten wird! Der NDR-Computer besteht aus kleinen Einheiten, die leicht erweitert oder ausgetauscht werden können. Damit sind Sie immer mit vorne dran!

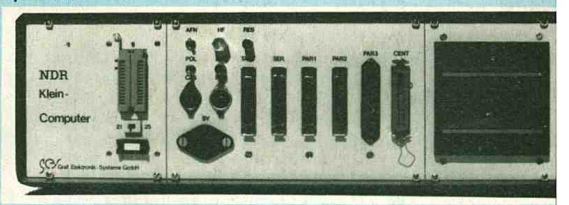
Der NDR hat sich für unseren Computer entschieden – tun Sie es auch!

Sie investieren in Ihre Zukunft – fordern Sie heute noch unsere ausführliche, kostenlose Info + Probeexemplar unserer Kundenzeitung an (bitte DM 1,40 Briefmarken für Rückporto beifügen).

Wie unterscheidet sich der NDR-Comput von Heimcomputern?

Der NDR-Computer ist teurer: aber nur auf den ersten Blick!

In der Grundausrüstung (z.B. Paket 1) ist der NDR-Com ter teurer als die gängigen Heimcomputer. Schon in dies Ausbau bietet er z.B. durch seine extrem schnelle, ho auflösende Graphik wesentlich mehr. Da der NDR-Com ter immer, auch später, erweitert oder umgebaut werd kann, z.B. auf eine andere Zentraleinheit, ist der ND Computer bald billiger als ein Heimcomputer, der in einig Jahren vielleicht nicht mehr dem Stand der Technik e spricht.



Der NDR-Computer ist Computer und Lehrmittel zugleich.

Durch den Selbstbau oder durch die Beschäftigung der umfangreichen Software lernen Sie mehr über Com ter als bei einem fertigen, abgeschlossenen System. Die Wissen über Computer wird heute und in der Zuk gefordert – vom Schüler bis zum Ingenieur, vom Techn bis zur Hausfrau.

Der NDR-Computer spricht deutsch und kennt keine Geheimnisse.

Alle Schaltpläne, Unterlagen und die Quellen aller P gramme sind frei verfügbar!

Vergleichen Sie – wir dürfen es nicht!

Graf Elektronik Systeme GmbH

Magnusstraße 13 · 8960 Kempten
Telefon (0831) 6211

Filiale München: Georgenstraße 61 · 8000 München 40 · Telefon (089) 2715
Filiale Hamburg: Ehrenbergstraße 56 · 2000 Hamburg 50 · Telefon (040) 388

e von uns aus 1. Hand!

Bitte benutzen Sie die Bestellkarte am Ende des Heftes!

nige Bestellbeispiele:

r haben die gängigsten Konfigurationen zu Paketangeboten zu-mmengefaßt. Die in den Paketangeboten angegebenen Preise nd Einzelpreise. – Besonders günstig ist unser Einsteigerpaket:

AKET E	**Einsteigerpaket für Anfänger geeignet**		
BC2	89,00	139,00	CPU
E	75,00	115,00	Ein-Ausgang
JS1	59,00	79,00	BUS
DW5V	59,00	85,00	Stromvers.
EXIO	189,00	279,00	Tastenfeld
HEX	30,00	30,00	Programm
EXIOH	30,00	30,00	Handbuch
	531,00	757,00	
eis	**398,00**	**695,00**	
AKET 1	**das Z80-Paket mit Tastatur**		
ONDERH	28,00	28,00	Sonderheft
BC2	89,00	139,00	CPU
E	75,00	115,00	Ein-Ausgang
EY	59,95	99,00	Für Tastatur
DP64K	379,00	468,00	Ausgabe
AS	74,90	129,00	Für Kassette
JS1	59,00	79,00	BUS
AST1	195,00	195,00	Tastatur
KAB	18,00	18,00	Kabel dazu
GRUND	60,00	60,00	Programm
	1037,85	1330,00	
eis	**949,00**	**1249,00**	
AKET 4	**für Einsteiger, die später zum CP/M-Rechner ausbauen wollen**		
PUZ80	89,00	139,00	VollCPU
DA64	39,95	84,00	64K
HEX2	30,00	30,00	Programm
JS3	169,00	209,00	Großer BUS
EXIOH	30,00	30,00	Handbuch
E2	295,00	295,00	Netzteil
IE	75,00	115,00	Ein-Ausgang
EXIO	189,00	279,00	Hex-Ein-Aus
	916,95	1181,00	
eis	**849,00**	**1145,00**	
AKET 6	**Der komplette CP/M-Computer mit 800K Floppy-Kapazität pro Diskette und einem Laufwerk**		
PUZ80	89,00	139,00	CPU
ANKBOOT	66,90	94,00	Boot-Karte
FLOMON	75,00	75,00	Monitor
R2	40,00	40,00	Rams
AM64/256	129,00	198,00	64K RAM
64DYN	129,00	129,00	mit Speicher
DP64K	379,00	468,00	Ausgabe
EY	59,95	99,00	Tastatur
AST2	395,00	395,00	Große Tast.
JS3	169,00	209,00	Großer BUS
E2	295,00	295,00	Netzteil
LO2	389,00	462,00	Floppy-C.
LACFD	698,00	698,00	Floppy-LW
AB008	38,00	38,00	Kabel
P/M2.2	398,00	398,00	Betriebssyst.
P/MDo	98,00	98,00	Doku dazu
	3447,85	3835,00	
eis	**3398,00**	**3800,00**	

e angegebenen Preise sind freibleibend (Stand Januar '85).
rdern Sie unsere aktuelle Preisliste an.

ges Aktuell

Jetzt lieferbar:

Betriebssystem CP/M 2.2
Kapazität 800 KByte pro Floppy
unterstützt RAM-Floppy 398.–

TEAC-Laufwerke FD55F, dazu passend .. 698.–

FLO2 Floppy-Disc-Controller-Karte,
für alle Formate, Bausatzpreis 389.–

Dynamische RAM-Karte 64 KByte od. 256 KByte
Bausatz (ohne RAMs) 129.–
64 KByte RAMs 129.–
256 KByte RAMs 560.–

Was wir für Sie tun:

Wir liefern alle Bausätze, Fertiggeräte, Programme und natürlich das gesamte Zubehör – wenn möglich ab Lager und alles aus einer Hand.

Wir helfen Ihnen vor und nach dem Kauf: durch ausführliche Beratung in unseren Filialen, durch eine günstige Pauschalreparatur, falls es mal wirklich nicht klappt.

Wir halten Sie auf dem Laufenden: Durch unsere eigene Kundenzeitschrift, die Sie über Neuentwicklungen, Anwendungen, Tips und Tricks informiert und die nur DM 10,00 für vier Ausgaben kostet.

Wir sorgen mit **Weiterentwicklungen** dafür, daß Sie immer ein modernes System nach dem neuesten Stand der Technik besitzen.

Besuchen Sie uns auch in unseren Filialen:

Ehrenbergstraße 56 · 2000 Hamburg 50 (Altona)

Georgenstraße 61 · 8000 München 40 (Schwabing)

Teilansicht unserer Filiale in München

Wir stellen aus:

Hobbytronic Dortmund

20. – 24. März '85

Mittwoch, 17. – Mittwoch, 24. April

Hannover Messe '85

Halle 12 / 2. OG / Stand 2321

ges

Rolf-Dieter Klein

Taschenrechner selbstgebaut

Ziel dieser Folge ist es, daß Sie mit Syntaxdiagrammen vertraut werden und sie als Programmierhilfsmittel verwenden können. Zum Beispiel soll ein Taschenrechner konstruiert werden. Neben den vier Grundrechenarten soll er auch die Klammertechnik beherrschen. Und natürlich die Rechenregeln, also Punkt vor Strich und eingeklammerte vor allen anderen Operationen.

Zur Realisierung wird hier nicht aus dem Pflichtenheft ein Struktogramm erstellt, denn das Problem soll rekursiv gelöst werden. Dafür gibt es andere Methoden zur Erarbeitung eines Programmes, zum Beispiel die Verwendung von Syntaxdiagrammen. *Bild 1* zeigt die Elemente, die zu einem Syntaxdiagramm zusammengesetzt werden können. Es gibt eine Flußrichtung und es gibt die Terminalsymbole, die, falls sie auftauchen, erkannt werden müssen. Weiter gibt es den Beschreibungsaufruf, wobei eine Beschreibung wieder aus einem Syntaxdiagramm besteht.

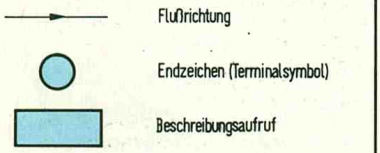


Bild 1. Die Elemente eines Syntaxdiagramms

Zum besseren Verständnis ein Beispiel. Die Definition eines Pascal-Namens zeigt *Bild 2*. Ein Name beginnt in Pascal grundsätzlich mit einem Buchstaben und besteht aus mindestens einem Buchstaben. Es können nach dem ersten Buchstaben weitere Buchstaben oder Ziffern folgen: HUBER, MEYER, MUELLER, B5HHY sind alles gültige Namen. In den Namen dürfen keine anderen Zeichen stehen als die angegebenen. Um *Bild 3* zu verstehen, muß man wissen, was Buchstaben und Ziffern sind. Dazu muß man die beiden Beschreibungsaufrufe ansehen. *Bild 3* zeigt die Definition eines Buchstabens und *Bild 4* die Definition einer Ziffer.

Mit Syntaxdiagrammen kann man sehr anschaulich Programmiersprachen beschreiben. So ist Pascal zum Beispiel vollständig mit Syntaxdiagrammen beschrieben. *Bild 5* zeigt als Beispiel die Definition der IF-Anweisung, so wie sie in Pascal benutzt wird. Die Anweisung beginnt mit dem Terminalsymbol IF. Danach muß ein Ausdruck folgen, der ein Entscheidungskriterium berechnet.

Dann folgt das Symbol THEN nachdem eine Anweisung steht. An dieser Stelle ist das Syntaxdiagramm schon rekursiv, denn die IF-Anweisung ist auch eine Anweisung. Folglich kann man nach THEN wieder eine IF-Anweisung stellen, so sagt es das Diagramm. Danach kann nun, wenn gewünscht, ein ELSE stehen, dem dann eine weitere Anweisung, der ELSE-Teil, folgen muß. So kann man alle Befehle von Pascal definieren. Es ist damit einfach, nachzuschauen, ob eine Programmkonstruktion erlaubt ist oder nicht. Genauso einfach ist es aber auch für den Programmierer des Compilers, diesen fehlerarm zu ge-

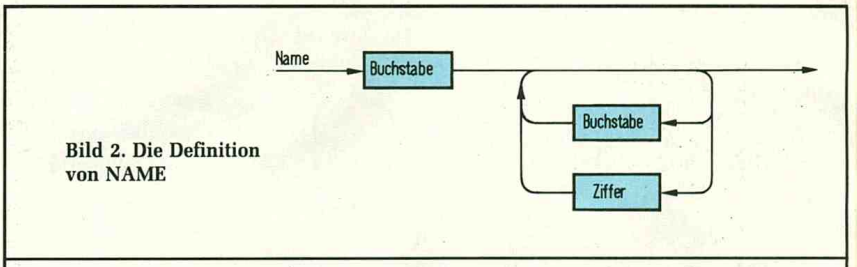


Bild 2. Die Definition von NAME

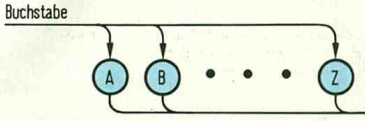


Bild 3. Die Definition von BUCHSTABE

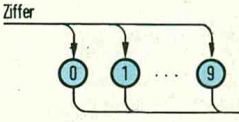


Bild 4. Die Definition von ZIFFER

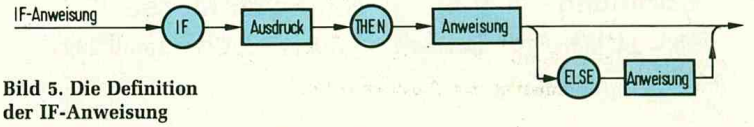


Bild 5. Die Definition der IF-Anweisung

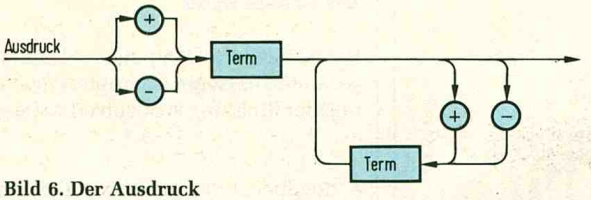


Bild 6. Der Ausdruck

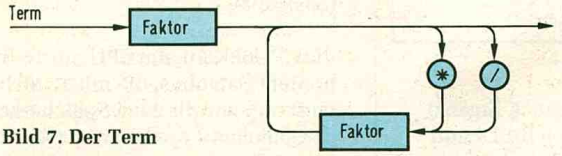


Bild 7. Der Term

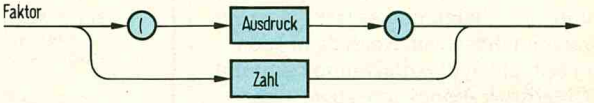


Bild 8. Der Faktor

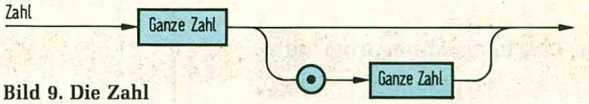


Bild 9. Die Zahl

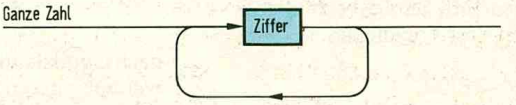


Bild 10. Die ganze Zahl

```
**********************************
* PASCAL/S  Pcode-Compiler V3.1 *
* (C) 1984 Rolf-Dieter Klein    *
* Version nach PASCAL/S von     *
* N.Wirth 1976, E.T.H Zuerich   *
**********************************

    0  program taschenrechner(input,output);
    0
    0  var
    0   ch : char;
    0   a : real;
    0
    0  procedure getchar;
    0   begin
    0    read(ch)
    2   end;
    2
    2
    2  procedure expr(var exprval : real);
    3   var nexttermval : real;
    3    ch1 : char;
    3
    3   procedure term(var termval : real);
    3   var nextfacval : real;
    3    ch2 : char;
    3
    3   procedure factor(var facval :real);
    3    var fac1val : real;
    3   begin
    3    if ch='(' then begin
    7     getchar;
   10     expr(facval);
   14     if ch = ')'
   16      then getchar
   18      else writeln(' Fehler ) fehlt ')
   24    end else begin
   26     facval:=0;
   30     while (ord(ch)>=ord('0')) and (ord(ch)<=ord('9')) do begin
   42      facval := facval * 10 + ord(ch) - ord('0');
   56      getchar;
   59     end;
   60     if ch='.' then begin
   64      fac1val:=10.0;
   67      getchar;
   70      while (ord(ch)>=ord('0')) and (ord(ch)<=ord('9')) do begin
   82       facval:=facval+(ord(ch)-ord('0'))/fac1val;
   94       fac1val:=fac1val*10;
  100       getchar;
  103      end;
  104     end;
  104    end;
  104   end;
  104
  104   begin (* term *)
  105    factor(termval);
  108    while (ch='*') or (ch='/') do begin
  116     ch2 := ch;
  119     getchar;
  122     factor(nextfacval);
  125     if ch2='*'
  127      then termval:=termval*nextfacval
  131      else termval:=termval/nextfacval
  137    end;
  141   end;
  141
  141   begin (* expr *)
  142    ch1 := ch;
  145    if (ch='+') or (ch='-') then getchar;
  156    term(exprval) ;
  159    if ch1='-' then exprval:=-exprval;
  167    while (ch='+') or (ch='-') do begin
  175     ch1 := ch;
  178     getchar;
  181     term(nexttermval);
  184     if ch1 = '+'
  186      then exprval := exprval + nexttermval
  190      else exprval := exprval - nexttermval
  196    end
  199   end;
  200
  200
  200
  200  begin (* Hauptprogramm *)
  201   writeln('Taschenrechner') ;
  204   repeat
  204    write(':');
  206    getchar;
  208    expr(a);
  211    writeln;
  212    writeln(' = ',a);
  217    writeln(' = ',a:14:8);
  224   until false
  224  end.
```

Bild 11. Programmlisting zum Taschenrechner

stalten, denn er kann sich am Syntaxdiagramm orientieren und es fast 1:1 in ein Programm umsetzen. Noch dazu kann man fast alle Syntaxdiagramme getrennt in Unterprogrammen umsetzen, ohne sich zunächst weiter um die Beschreibungsaufrufe zu kümmern. Denn diese werden ebenfalls isoliert oder fast isoliert umgesetzt.

Das Taschenrechnerprogramm

Bild 6 zeigt die Definition eines Ausdrucks, wie ihn unser Taschenrechner verstehen soll. Ein Ausdruck kann wiederum ein Term sein, der ein Vorzeichen haben darf, gefolgt von einem weiteren Term mit Vorzeichen.

Beispiele: –TERM,
TERM1+TERM2–TERM3.

Ähnlich sieht die Definition eines Terms aus. *Bild 7* zeigt das Syntaxdiagramm. In Faktor (*Bild 8*) schließlich wird wieder die Definition von Ausdruck verwendet.

Alles bisher genannte kann in Faktor wieder benutzt werden. Wenn nicht, dann ist ein Faktor einfach eine Zahl.

Damit sind die vier Grundrechenarten in Diagrammen definiert, wie auch die Klammerrechnung. Wo aber sind die Rechenregeln? Sie stecken in der Anordnung der Syntaxdiagramme. Denn bei einem Ausdruck werden zunächst die Terme ausgewertet und erst zuletzt addiert oder subtrahiert: Punkt vor Strich. Sind dann in einzelnen Faktoren Klammerausdrücke, so werden auch diese zunächst vollständig berechnet. Ergänzt werden müssen jetzt noch *Bild 9* und *Bild 10*, in welchen die Definition einer Zahl getroffen wird.

```
Start PCODE-Interpreter 3.1
Taschenrechner
:3*5+1.2/0.3

 =     1.900000000000e+01
 =    19.00000000
:0.5*1.2/4*(7-10)

 =    -4.500000000000e-01
 =    -0.45000000
:0.001*100000

 =     1.000000000000e+02
 =   100.00000000
:
```

Bild 12. So wird gerechnet

Die Umsetzung kann man am besten am fertigen Pascal-Programm studieren. *Bild 11* zeigt das Listing und *Bild 12* ein Eingabebeispiel. Bei der Realisation gibt es eine Besonderheit. Es werden zunächst die Prozeduren EXPR, TERM und FACTOR angegeben und dann erst folgt der zugehörige Programmteil in umgekehrter Reihenfolge. Dies ist in Pascal so, weil man Unterprogramme nur dann aufrufen darf, wenn man sie schon definiert hat. Bei rekursiven Programmen, die sich gegenseitig aufrufen, kann das zu Problemen führen. Man muß sie dann ineinander verschachteln, um die Schwierigkeit zu umgehen. Hier sei nochmals auf die Pascal-Literatur verwiesen.

Aufgaben

1. Welchen einfachen Fehler enthält die Programmrealisation im Programmteil FAKTOR? Vergleichen Sie dazu Syntaxdiagramm und Programmstruktur. Zeichnen Sie das Syntaxdiagramm so, daß es dem Programm entspricht.

2. Welches Ergebnis liefert der Taschenrechner nach der Eingabe „3+." (siehe auch Aufgabe 1)?

3. Erweitern Sie das Syntaxdiagramm für die Eingabe von Gleitkommazahlen mit Exponenten und programmieren Sie das Syntaxdiagramm.

4. Erweitern Sie den Taschenrechner um trigonometrische Funktionen.

5. Erweitern Sie den Taschenrechner um eine Speicherfunktion. Dazu sollen Variable mit den Namen A bis Z verwendet werden können. Zum Setzen einer Variablen soll man z. B. =A schreiben dürfen. Konstruieren Sie zuerst das Syntaxdiagramm und dann das komplette Programm.

Der weitere Ausbau

Der NDR-Klein-Computer ist kein fertiges Gerät, er ist ein Computersystem, das in jeder Richtung ausbaubar ist. Da gibt es

- die Speicherkarte DYN64/256, die bis zu 256 KByte dynamischen Speicher faßt,
- den Festplattencontroller FESTCON (Sasi-Bus),
- die CPU68K/16, die CPU mit 16 Bit breitem Datenbus, die mit 12 MHz läuft und jeweils zwei Speicherkarten des gleichen Typs benötigt (Wortbreite = 2 Byte)
- die CPU mit dem 80286, damit Sie auch mit IBM mithalten können,
- die CPU 32032, die aus Ihrem Rechner einen echten 32-Bit-Mini macht. Sie benötigen für den Speicherausbau in Wortbreite jeweils vier Speicherkarten.

Außerdem wird es Farbkarten geben, die Ihnen den Aufbau farbiger Bilder auf dem Bildschirm ermöglichen, wenn Sie einen RGB-Monitor besitzen. Die angekündigten Geräte existieren zum Zeitpunkt der Drucklegung alle in Laborversionen. Der Zeitpunkt, zu dem sie auf dem Markt erhältlich sind, hängt von den Problemen ab, die bei der Umsetzung in fertigungsreife Platinen noch auftreten können. Deshalb sollten Sie vor einer Bestellung anfragen, was alles schon lieferbar ist.

Was aber besonders wichtig ist: Der NDR-Computer führt kein isoliertes Dasein! Denn wir von mc haben uns bemüht, ein einheitliches Floppy-Disk-Aufzeichnungsformat für den berühmten mc-CP/M-Computer (mit neuem 80-Spur-BIOS), für den mc-68000-Computer (ein „echter" 16-Bit-Rechner mit CP/M-68k) und natürlich für das Floppy-Interface FLO-2 des NDR-Klein-Computers zu verwenden.

Und sicher ist Ihnen längst klar, daß der NDR-Klein-Computer nicht etwa so heißt, weil er besonders klein wäre, sondern nur deshalb, weil er von Rolf-Dieter-Klein für die NDR-Fernsehserie „Mikroelektronik" entwickelt wurde.
Ein Computer also, der nicht nur ausbaufähig, sondern auch didaktisch konzipiert ist.

Übersicht über den Befehlssatz des 68008

ABCD.B Dx,Dy	XNZVC	Addition von BCD-Zahlen
ABCD.B -(An),-(An)	XNZVC	Addition von BCD-Zahlen
ADD.x <ea>,Dn	XNZVC	Binäre Addition
ADD.x Dn,<ea>	XNZVC	Binäre Addition
ADDA.x <ea>,An	-	Binäre Addition mit Adressregister als Ziel
ADDI.x #konst,<ea>	XNZVC	Binäre Addition mit Konstante als Quellop.
ADDQ.x #konst,<ea>	XNZVC	Addiere schnell (Werte 1..8)
ADDX.x Dx,Dy	XNZVC	Addiere mit Extendbit
ADDX.x -(Ax),-(Ay)	XNZVC	Addiere mit Extendbit
AND.x <ea>,Dn	-NZVC	Logische Und-Verknüpfung
AND.x Dn,<ea>	-NZVC	Logische Und-Verknüpfung
ANDI.x #konst,<ea>	-NZVC	Logische Und-Verknüpfung mit einer Konst.
ANDI #konst,CCR	XNZVC	Und-Verknüpfung mit Condition-Code-Reg.
ANDI #konst,SR	XNZVC	Und-Verknüpfung mit Statusregister (priv)
ASL.x Dx,Dy	XNZVC	Links Schieben indirekt
ASL.x #konst,Dn	XNZVC	Links Schieben, um "konst" Bits
ASL <ea>	XNZVC	Links Schieben, um Eins
ASR.x Dx,Dy	XNZVC	Rechts Schieben indirekt
ASR.x #konst,Dn	XNZVC	Rechts Schieben, um "konst" Bits
ASR <ea>	XNZVC	Rechts Schieben, um Eins
Bcc marke	-	Bedingter Sprung, 16-Bit-Adressdistanz
Bcc.S marke	-	Bedingter Sprung, 8-Bit-Adressdistanz
BCHG Dn,<ea>	--Z--	Prüfe ein Bit und komplementiere es
BCHG #konst,<ea>	--Z--	Prüfe ein Bit und komplementiere es
BCLR Dn,<ea>	--Z--	Prüfe ein Bit und setze es auf Null
BCLR #konst,<ea>	--Z--	Prüfe ein Bit und setze es auf Null
BRA marke	-	Sprungbefehl, 16-Bit-Adressdistanz
BRA.S marke	-	Sprungbefehl, 8-Bit-Adressdistanz
BSET Dn,<ea>	--Z--	Prüfe ein Bit und setze es auf Eins
BSET #konst,<ea>	--Z--	Prüfe ein Bit und setze es auf Eins
BSR marke	-	Unterprogrammaufruf, 16-Bit-Adressdistanz
BSR.S marke	-	Unterprogrammaufruf, 8-Bit-Adressdistanz
BTST Dn,<ea>	--Z--	Prüfe ein Bit
BTST #konst,<ea>	--Z--	Prüfe ein Bit
CHK <ea>,Dn	-NZVC	Prüfe gegen Grenzbereich
CLR.x <ea>	-NZVC	Löschen den Inhalt eines Operanden
CMP.x <ea>,Dn	-NZVC	Vergleich zweier Operanden
CMPA.x <ea>,An	-NZVC	Vergleich mit einem Adressregister
CMPI.x #konst,<ea>	-NZVC	Vergleich einer Konstanten mit Operand
CMPM.x (Ax)+,(Ay)+	-NZVC	Vergleich zweier Speicherzellen
DBcc Dn,marke	-	Prüfe Bedingung, decrementiere und spinge
DBRA Dn,marke	-	Decrementiere und springe
DIVS <ea>,Dn	-NZVC	Division mit Vorzeichen
DIVU <ea>,Dn	-NZVC	Division ohne Vorzeichen
EOR.x Dn,<ea>	-NZVC	Logische Exklusiv-Oder-Verknüpfung
EORI.x #konst,<ea>	-NZVC	Logische Exklusiv-Oder-Verknüpfung
EORI #konst,CCR	XNZVC	Exklusiv-Oder mit Condition-Code-Register
EORI #konst,SR	XNZVC	Exklusiv-Oder mit Status-Register (priv)
EXG Rx,Ry	-	Vertausche Registerinhalte
EXT.x Dn	-NZVC	Vorzeichenrichtige Erweiterung
ILLEGAL	-	Unzulässiger Befehl, führt zur Exception
JMP <ea>	-	Springe absolut
JSR <ea>	-	Unterprogrammaufruf absolut
LEA <ea>,An	-	Lade effiktive Adresse in Adressregister
LINK An,#konst	-	Rette A7 und lege neuen Stackbereich an
LSL.x Dx,Dy	XNZVC	Logische Verschiebung nach Links, indirekt
LSL.x #konst	XNZVC	Logische Verschiebung nach Links
LSL.x <ea>	XNZVC	Logische Verschiebung nach Links
LSR.x Dx,Dy	XNZVC	Logische Verschiebung nach Rechts, indirekt
LSR.x #konst	XNZVC	Logische Verschiebung nach Rechts
LSR.x <ea>	XNZVC	Logische Verschiebung nach Rechts
MOVE.x <ea>,<ea>	-NZVC	Übertrage Daten von Quelle nach Ziel
MOVE <ea>,CCR	XNZVC	Übertrage ein Datum in Condition-Code-Reg.
MOVE <ea>,SR	XNZVC	Übertrage ein Datum in Status-Reg. (priv)
MOVE SR,<ea>	XNZVC	Übertrage Status-Reg. in Zieloperand
MOVE USP,An	-	Anwenderstackpointer nach An (priv)
MOVE An,USP	-	An nach Anwenderstackpointer (priv)
MOVEA.x <ea>,An	-	Übertrage Wert in Adressregister
MOVEM.x reglist,<ea>	-	Register nach Ziel speichern
MOVEM.x <ea>,reglist	-	Quelle nach Register
MOVEP.x Dx,d(Ay)	-	Inhalt von Dx nach Adresse d(Ay)
MOVEP.x d(Ay),Dx	-	Inhalt von Adresse d(Ay) nach Dx
MOVEQ #konst,Dn	-	Laden einer Konstanten, schnell
MULS <ea>,Dn	XNZVC	Multiplikation mit Vorzeichen
MULU <ea>,Dn	XNZVC	Multiplikation ohne Vorzeichen
NBCD <ea>	XNZVC	Negation von BCD-Zahlen
NEG.x <ea>	XNZVC	Negation
NEGX.x <ea>	XNZVC	Negation mit Berücksichtigung des X-Flags
NOP	-	Keine Operation (Nichtstubefehl)
NOT.x <ea>	XNZVC	Logisches Komplement
OR.x <ea>,Dn	-NZVC	Logische Oder-Verknüpfung
OR.x Dn,<ea>	-NZVC	Logische Oder-Verknüpfung
ORI.x #konst,<ea>	-NZVC	Logische Oder-Verknüpfung mit Konstanten
ORI #konst,CCR	XNZVC	Logisches Oder mit Condition-Code-Register
ORI #konst,SR	XNZVC	Logisches Oder mit Status-Register (priv)
PEA <ea>	-	Lege die effektive Adresse auf dem Stack ab
RESET	-	Rücksetzen externen Peripherie (priv)
ROL.x Dx,Dy	XNZVC	Rotiere links, indirekt
ROL.x #konst,Dy	XNZVC	Rotiere links, um Anzahl "konst"
ROL.x <ea>	XNZVC	Rotiere rechts, um Eins
ROR.x Dx,Dy	XNZVC	Rotiere rechts, indirekt
ROR.x #konst,Dy	XNZVC	Rotiere rechts, um Anzahl "konst"
ROR.x <ea>	XNZVC	Rotiere rechts, um Eins
ROXL.x Dx,Dy	XNZVC	Rotiere links, indirekt mit X-Flag
ROXL.x #konst,Dy	XNZVC	Rotiere links, um Anzahl "konst" mit X-Flag
ROXL.x <ea>	XNZVC	Rotiere rechts, um Eins mit X-Flag
ROXR.x Dx,Dy	XNZVC	Rotiere rechts, indirekt mit X-Flag
ROXR.x #konst,Dy	XNZVC	Rotiere rechts, um Anzahl "konst" mit X-Flag
ROXR.x <ea>	XNZVC	Rotiere rechts, um Eins mit X-Flag
RTE	XNZVC	Rückkehr aus Exception (priv)
RTR	XNZVC	Rückkehr mit Laden der Flags
RTS	-	Rückkehr aus dem Unterprogramm
SBCD Dx,Dy	XNZVC	Subtraktion von BCD-Zahlen
SBCD -(Ax),-(Ay)	XNZVC	Subtraktion von BCD-Zahlen
Scc	XNZVC	Setze ein Byte je nach Bedingung
STOP #konst	XNZVC	Lade Statusregister und HALT (priv)
SUB.x <ea>,Dn	XNZVC	Binäre Subtraktion
SUB.x Dn,<ea>	XNZVC	Binäre Subtraktion
SUBA.x <ea>,An	-	Binäre Subtraktion ins Adressregister
SUBI.x #konst,<ea>	XNZVC	Binäre Subtraktion mit einer Konstanten
SUBQ.x #konst,<ea>	XNZVC	Subtrahiere schnelle (1..8)
SUBX.x Dx,Dy	XNZVC	Subtrahiere mit Extendbit
SUBX.x -(Ax),-(Ay)	XNZVC	Subtrahiere mit Extendbit
SWAP Dn	-NZVC	Tausche Inhalt von Registerhälften
TAS <ea>	-NZVC	Prüfe und setze ein Bit im Zieloperanden
TRAP #konst	-	Trap-Befehl
TRAPV	-	Trap bei Überlauf
TST.x <ea>	-NZVC	Prüfe den Inhalt eines Operanden
UNLK An	-	Hole den Stackpointer zurück

Pseudobefehle des Assemblers:

DC.x werteliste	Die durch Komma getrennten Werte der Werteliste werden im Speicher abgelegt. Dabei sind bei DC.B auch Texte möglich, die zwischen zwei einfachen Anführungszeichen gesetzt werden. Beispiel: DC.B 1,'Hallo',0 DC.W $AA55 DC.L alpha, 345, geht-4
DS.x wert	Ein Speicherbereich wird reserviert. Dorthin werden keine Daten abgelegt. Beispiel : DS.W 100
DS 0	Falls der Programmzähler auf einer ungeraden Adresse steht, wird ein Byte in das Programm eingefügt, sonst nicht. Die nächste freie Speicherzelle liegt damit immer auf einer geraden Adresse.
END	Ende des Assemblerprogramms. Aller folgender Text in der Quelle wird ignoriert.
symbol EQU wert	Der Wert des Ausdrucks "wert" wird dem Symbol zugewiesen. Jedes Symbol darf aber nur eine Zuweisung erhalten. Beispiel: HUNDERT EQU 100
OFFSET wert	Wird zur Ablageadresse addiert, ohne jedoch bei Marken Einfluß zu haben. Damit kann man Code z.B. für die Adresse 0 erzeugen und dennoch im RAM ablegen. Beispiel: OFFSET $9C00 ORG 0
ORG wert	Startadresse für die Ablage des Codes. Wenn kein OFFSET angeben wurde ist dies die Speicheradresse von wo ab das folgende Assemblerprogramm, oder Daten abgelegt wird. Beispiel: ORG $A000

Verwendete Abkürzungen:

cc	Bedingungscode (CC,CS,EQ,GE,GT,HI,LE,LS,LT,MI,NE,PL,VC,VS, T,F)
.x	Wortbreite (.B,.W,.L) bei Adressregistern ist nur .W und .L erlaubt)
reglist	Registerbereichsangabe, z.B. D0-D3/A1/A3-A6
konst	Konstante, dies kann auch ein komplexer Ausdruck mit Symbolen und arithmetischen Operationen sein
Dn,Dx,Dy	Angabe eines Datenregisters D0,D1...D7
An,Ax,Ay	Angabe eines Adressregisters A0,A1...A7
Rx,Ry	Daten oder Adressregister
marke	Zieladresse, oder Ausdruck
<ea>	Effektive Adresse:

Dn	Datenregister
An	Adressregister
(An)	indirekte Adresse
(An)+	mit Post-Increment
-(An)	mit Pre-Decrement
d(An)	mit Displacement
d(An,Rx)	mit Displacement und Index
wert.W	absolute Adresse 16 Bit
wert.L	absolute Adresse 32 Bit
d(PC)	PC-Relativ
d(PC,Rx)	mit Index
#konst	direkter Wert

Nicht alle Adressierarten sind bei allen Befehlen mit der <ea>-Angabe zulässig. Als Ziel ist z.B. der direkte Wert nicht erlaubt, als Ziel auch nicht die PC-Relativ-Adressierart.

d,marke,konst wert	Es sind immer komplexe Ausdrücke mit Symbolen und arithmetischen Operationen erlaubt.

Die Befehle des 68008-Grundprogramms in Kürze

Nummer	Name	Funktion	Parameter
1	Schreite	Bewege die Schildkröte in Blickrichtung	D0 Schrittzahl
2	Drehe	Drehe die Schildkröte entgegen dem Uhrzeiger	D0 Winkel
3	Hebe	Die Schildkröte hinterläßt nun keine Spur mehr	
4	Senke	Nun wieder Schreibspur	
5	-	undefiniert	
6	-	undefiniert	
7	Set	Setze die Schildkröte an eine bestimmte Stelle	D1 X-Koord., D2 Y-Koord., D3 Blickrichtung
8	Moveto	Setzen der Zeichen- oder Schreibposition	D1 X-Koord., D2 Y-Koord.
9	Drawto	Zeichne Linie nach X,Y	D1 X-Pos., D2 Y-Pos.
10	Write	Texte auf dem Bildschirm ausgeben A0 Adresse des Textpuffers	D0 Textgröße = dx*16+dy D1 X-Koord., d2 Y-Koord.
11	Read	Eingabe von Text im Rahmen wie im Grundprogramm	D0 Größe (wie Write) D1 X-Koord., D2 Y-Koord., D3 Max. Zahl der Zeichen, A0 Adresse des Textpuffers
12	Ci	Eingabe eines Zeichens von Tastatur	D0 Empfangenes Zeichen
13	Csts	Tastaturabfrage (Status) D0=0, wenn keine Taste gedrückt wurde	D0 Status
14	Ri	Lesen eines Zeichens von der CAS-Schnittstelle	D0 Zeichen
15	Po	Ausgeben eines Zeichens auf die CAS-Schnittst.	D0 Zeichen
16	Clr	Löschen aller vier Bildseiten	
17	Clpg	Löschen einer unsichtbaren Bildseite (s. Newpage)	
18	Wait	Warten bis der GDP-Prozessor fertig ist	
19	Schrittel	Schreite um 1/16 Bildpunkt (z.B. für Kreise)	D0 Anzahl Schritte
20	Clrscreen	Löschen Bildschirm und vorbereiten für die Textausgabe (s. Co)	
21	Co	Textausgabe (40/80 Zeichen pro Zeile) Verschiedene ASCII-Steuercodes werden ausgewertet	D0 Zeichen
22	Lo	Ausgabe eines Zeichens auf Drucker (Centronics)	D0 Zeichen
23	Sin	Berechnet Sinus. -256 <= Ergebnis <= +256	D0 Winkel in Grad, gleichzeitig Ergebnis
24	Cos	Berechnet Cosinus (s. Sin)	D0 Winkel/Ergebnis
25	Size	Zeichengröße für Co und Co2 ($11 = 80 Zeichen, $21 = 40 Zeichen)	D0 Zeichengröße

Nummer	Name	Funktion	Parameter
26	Cmd	Direktbefehl an den GDP-Prozessor (s. Wait und GDP-Datenblatt)	D0 Befehlscode
27	Newpage	Wahl der Bildseite	D0 Schreibseite, D1 Leses.
28	Sync	Abfrage der Vertikal-Synchronpulse des GDP (für flimmerfreies Bild. D0 <>0 --> Sync-Puls)	D0 Sync-wert
29	Wert	Auswerten eines Ausdrucks (siehe Menübeschreibung)	A0 Adresse Textpuffer D0 Ergebnis, d1 Fehlercode
30	Zuweis	Zuweisung Symbolname:=Wert	A0 Textpuffer mit Ausdruck
31	Ciinit2	Setzen Textzeiger auf Textanfang (für Ci2)	
32	Ci2	Texte aus dem Speicher lesen (statt von Tastatur)	D0 Zeichen
33	Co2	Erweiterter Co (mit Ausgabeumlenkung und Fluchtsymbolen --> siehe Handbuch)	D0 Zeichen
34	Setflip	Rate für die Bildseitenumschaltung	D0 Rate Zweiseitenumsch., D1 Rate Vierseitenumsch.
35	Delay	Zeitverzögerung im 100 ms-Raster	D0 Zeit in 1/10 sec
36	Firsttime	Initialisierung der Schildkrötengrafik	
37	Setpen	'Schreibstift' senken	
38	Erapen	'Schreibstift' in Löschmodus	
39	Grapoff	Ausschalten der Schildkrötengrafik	
40	Cmdprint	Befehlssequenz an GDP geben (Text oder Kurzvektoren)	D0 Textgröße, A0 Textpuffer, D1 X-Koord., D2 Y-Koord.
41	Print2x	Sedezimale Ausgabe eines Bytes in einen Puffer	D0 Wert, A0 Pufferadresse
42	Print4x	dito für Wort	
43	Print6x	dito 6-stellig	
44	Print8x	dito 8-stellig	
45	Print8b	Binäre Ausgabe eines Bytes in einen Puffer	D0 Wert, A0 Pufferadresse
46	Print4d	4-stell. dezimale Ausgabe e. Wertes in e. Puffer	D0 Wert, A0 Pufferadresse
47	Hide	Abschalten der Schildkröte	
48	Show	Einschalten der Schildkröte	
49	Crt	Co2-Ausgabe auf Bildschirm legen	
50	Lst	Co2-Ausgabe auf Drucker legen	
51	Usr	Co2-Ausgabe auf Benutzerroutine legen (Vektor in $8024)	
52	Nil	Co2-Ausgabe in ein 'schwarzes Loch' umleiten	
53	Seterr	ERRFLAG setzen (löscht Nil, falls ERRFLG <> 0)	D0 Fehlerflag

Nummer	Name	Funktion	Parameter
54	Geterr	Wert von ERRFLG lesen	D0 Fehlerflag (Erg.)
55	Setpass	Setzen PASSFLAG (falls <> 2, wird Co2 unterdr.)	D0 Flagwert
56	Edit	Aufruf des Editors	
57	Figur	Figurbefehl (siehe Beschreibung in diesem Heft)	D0 Größe, A0 Adresse der Figurdefinition, D1 X-Koord., D2 Y-Koord.
58	Setfig	Fixieren eines mit Figur gezeichneten Bildes	
59	Getram	Suche Ram-Bereich (1K-Byte-Schritte) (nach Aufruf: A1 Start Ram, A0 Ende Ram+1)	A0 Startadresse f. Suche
60	Autoflip	Bildseitenumschaltung	
61	Cursein	Einschalten des Textcursors	
62	Cursaus	Abschalten des Textcursors (wichtig vor Co2)	
63	Char	Wie Co, jedoch ohne Steuerzeichen (nicht verwenden)	D0 Zeichen
64	Progzge	Zeichengenerator Programmieren (Zeichenpuffer 8 x 6 Zeichen = 6 Byte)	A0 Zeichendefinition
65	Assemble	Aufruf Assembler	
66	Getstx	Start des Editorpuffers	D0 Adresse
67	Putstx	Setzen Textpufferadresse (Textende --> Nullbyte)	D0 Adresse
68	Getorg	Default-Programmzählerwert holen	D0 Wert
69	Putorg	Setzen Default-Programmzähler	D0 Wert
----------		Die folgenden Befehle sind in Version 4.3 neu hinzugekommen	----------
70	Print8d	Dezimale Ausgabe einer 32-Bit-Größe ohne Vorz.	D0 Wert, A0 Pufferadresse
71	Printv8d	dito mit Vorzeichen	
72	Muls32	32-bit-Multiplikation	D1,D0 := D0 * D2
73	Divs32	32-bit-Division (D1 mit 0 vorbesetzen)	D0 := D0 / D2, D1 := REST
74	Flinit	Initialisieren Floppy-Software	
75	Floppy	Floppy-Routine für Schreiben, Lesen, Steprate (näheres siehe Handbuch)	A0 Pufferadresse, D1 Befehl. D2 Sektor, D3 Spur, D4 Laufwerk und Dichte
76	Getflop	Ermitteln der Schreibdichte einer Floppy	D4 Laufwerk/zur. m. Dichte
77	Setxor	Setzen zweite Hälfte des Seitenreg. des GDP (erfordert Schaltungsänderung, s. HB)	D0 Wert (0..F)
78	Getxor	Rücklesen des o.g. Wertes	D0 Wert
79	Setcolor	Setzen Farbe (für Erweiterungen)	D0 Farbe
80	Getcolor	Rücklesen Farbe	D0 Wert
81	Curon	Cursor anzeigen bei Co, Co2	
82	Curoff	Keinen Cursor anzeigen bei Co, Co2	
83	Adj360	Winkel in D0 auf den Bereich 0...359 bringen	D0 Winkel
84	Prtsym	Ausgabe der Symblotabelle mit Co2	
85	Symclr	Symboltabelle löschen	
86	Getsym	Adresse der Symboltabelle	D0 Adresse, A0 Adresse
87	Getnext	Länge der Symboltabelle (0...$FFFF)	D0 Länge
88	Putnext	Ändern der Symboltabellenlänge (zum Speichern von Variablen und dergleichen)	D0 neue Länge
89	Getbasis	aktuelle Anfangsadresse des Grundprogramms (das Grundprogramm ist ab Vers. 4 relokativ)	D0 Adresse, A0 Adresse
90	Getvar	Variablenbereich des Grundprogramms (default $8000)	D0 Adresse, A0 Adresse
91	SetA5	Setzen A5 auf Getvar (A5 ist reserviert)	
92	Aufxy	Setzen der Schildkröte	D1 X-Koord., D2 Y-Koord.
93	Korxy	Ermitteln der Schildkröten-Koordinaten	D1 X-Koord., D2 Y-Koord.
94	Aufk	Setzen der Schildkröten-Blickrichtung	D0 Winkel
95	Getk	Ermitteln des Schildkröten-Winkels	D0 Winkel
96	Rnd	Zufallszahl zwischen 0 und n-1 (D0 enthält beim Aufuf n und nacher die Zufallsz.)	D0 n/Zufallszahl
97	Getvers	Versionsnummer des Grundprogramms (dreistellig)	D0 Versionsnummer
98	Getsn	Seriennummer aus Adresse $410 holen (32-Bit)	D0 Seriennummer
99	Crlf	Ausgabe von Carriage Return und Linefeed über Co2	
100	Getline	Zugriff auf eine Textzeile des Bildschirms (in A2 steht nach dem Aufruf die Cursorspalte)	D0 Zeilennummer (0..23), A0 Pufferadresse
101	Getcurxy	Ermitteln der Cursorposition	D1 X-Pos., D2 Y-Pos.
102	Setcurxy	Setzen des Cursors	D1 X-Koord., D2 Y-Koord.
103	Getxy	Ermitteln der aktuellen Zeichenposition	D1 X-Pos., D2 Y-Pos.
104	Si	Eingabe eines Zeichens von der seriellen Schnittst.	D0 Zeichen
105	So	Ausgabe eines Zeichens auf die serielle Schnittst.	D0 Zeichen
106	Sists	Status der seriellen Eingabe (analog Csts)	D0 Status
107	Sosts	Status des seriellen Ausgabepuffers	D0 Status
108	Siinit	Einstellen der seriellen Schnittstelle	D0 Control-R., D1 Command-R.
----------		Die folgenden Befehle beziehen sich auf geplante Hardwareerweiterungen	----------
109	Getad8	Lesen A/D-Umsetzer AD 8x16	D0 Kanal, Wert zurück
110	Getad10	Lesen A/D-Umsetzer AD10x1	D0 Wert
111	Setda	Setzen Wert des D/A-Wandlers DA 8x2	D1 Kanal 1, D2 Kanal 2
112	Speak	Ausgabe auf Sprachsynthesizer mit dem SC-02 (SSI 236), das erste Wort des Parameter Blocks enthält seine Länge	A0 Adresse Parameterblock
113	Speak1	Ausgabe eines einzelnen Phonems	A0 Adresse Parameterblock
114	Sound	Bedienen der Soundgeneratorplatine	A0 Adresse Parameterblock
115	Getuhr	Bedienen der Uhren-Baugruppe	A0 Pufferadresse
116	Setuhr	Stellen der Uhr	A0 Pufferadresse
117	Lststs	Status des Druckers (Centronics) (z.B. für Spooler)	D0 Status
118	Relan	Relais auf der CAS-2-Baugruppe einschalten	
119	Relaus	Relais ausschalten	
120	Asserr	Ermitteln der Anzahl Assemblerfehler	D0 Fehlerzahl

Rolf Dieter Klein

Die Z80-CPU voll ausgebaut

Bisher haben Sie in diesem Kurs nur einen kleinen, aber vollständigen Z80-Computer kennengelernt: die Karte SBC2. Jetzt soll ein Z80-Rechner mit allem Komfort aufgebaut werden, der mit Floppys und allem, was einen richtigen Mikrocomputer auszeichnet, ausgerüstet werden kann.

Wenn Sie die SBC2-Karte aufgebaut haben, dann werden Sie festgestellt haben, daß darin die Fähigkeiten des Z80 nicht vollständig ausgenutzt wurden. Vor allem waren die Speicheradressen beschränkt, die der CPU für ihre Arbeit zur Verfügung standen. Das liegt daran, daß diese Karte besonders preiswert sein sollte, damit jedermann zunächst einmal einen vollständigen Computer in Besitz nehmen kann. Wenn jemand danach feststellt, daß die Sache mit den Computern wirklich Spaß macht, dann kann und muß er mit einem neuen Konzept professionell einsteigen, während jemand ohne Vorliebe für Computer nur wenig investiert hat und ohne Reue und mit vielen wertvollen Kenntnissen versehen, jetzt aussteigen kann.

Ein modulares Konzept

Betrachten Sie *Bild 1*. Es zeigt ein Schema, das alle Z80-Signale und noch einiges mehr enthält. Es ist die logische Grundlage des NDR-Klein-Computersystems, die vor allem die nahezu beliebige Ausbaubarkeit des Gerätes garantiert. Im Grunde ist es nur eine Beschreibung eines Leitungspaketes, das auf einer Platine (wenn es die Normalausführung ist) sechs Steckplätze untereinander verbindet. Diese Platine haben Sie schon kennengelernt, es ist die Busplatine, die auch das physikalisch tragende Element in diesem Computersystem bildet. Die SBC2-Platine hat nur etwa die Hälfte dieser Leitungen ausgenutzt, von denen bis auf zwei Ausnahmen alle parallel von Sockel zu Sockel laufen.

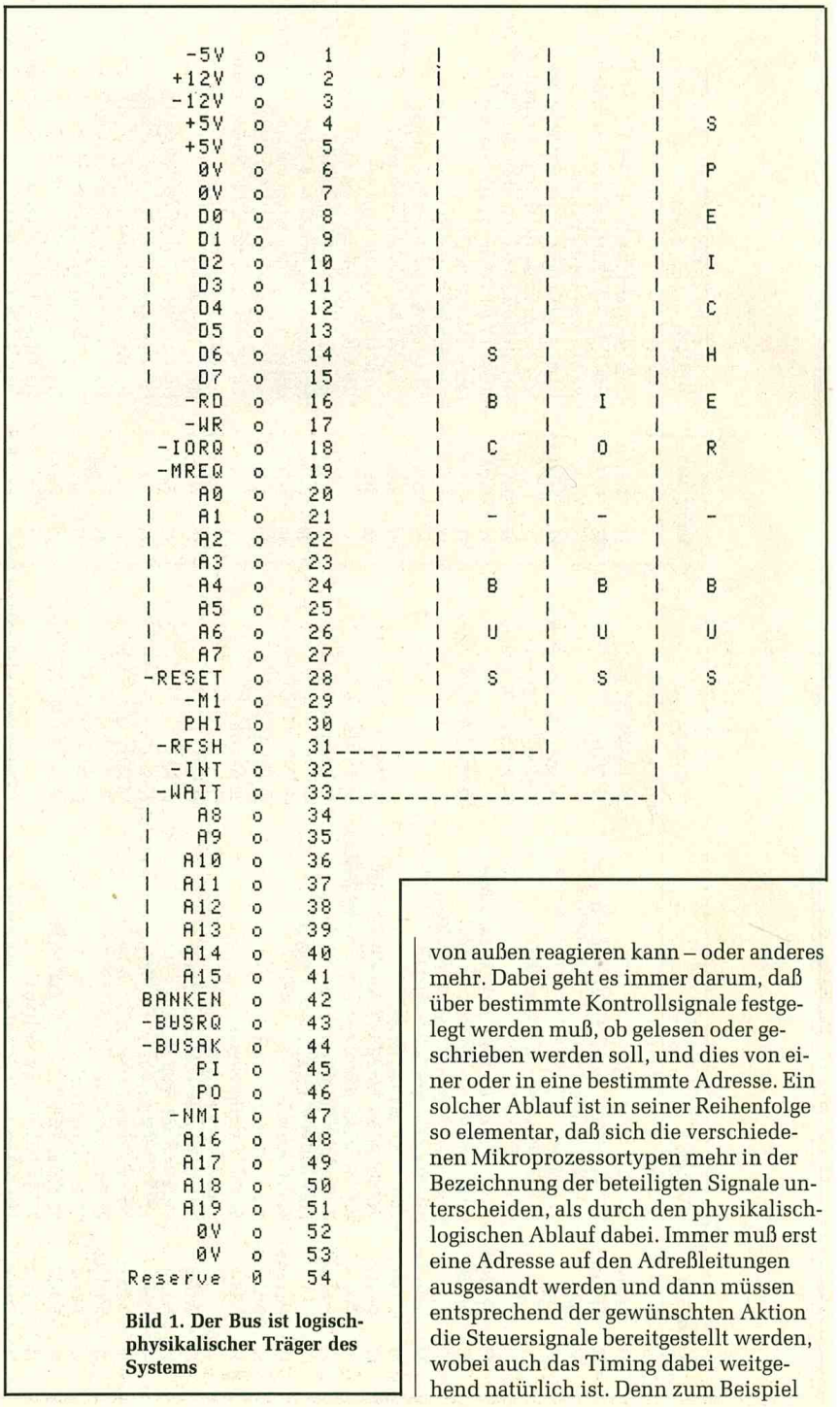


Bild 1. Der Bus ist logisch-physikalischer Träger des Systems

Der Zweck eines solchen Bussystems ist einfach zu beschreiben. Da im Anwendungsfeld eines Computers immer wieder neue Anforderungen auftreten können, muß ein ordentlicher Computer möglichst einfach erweiterbar sein, damit er zum Beispiel mehr Speicher bekommen kann, oder etwa eine Platine, mit der er besonders schnell auf Signale von außen reagieren kann – oder anderes mehr. Dabei geht es immer darum, daß über bestimmte Kontrollsignale festgelegt werden muß, ob gelesen oder geschrieben werden soll, und dies von einer oder in eine bestimmte Adresse. Ein solcher Ablauf ist in seiner Reihenfolge so elementar, daß sich die verschiedenen Mikroprozessortypen mehr in der Bezeichnung der beteiligten Signale unterscheiden, als durch den physikalisch-logischen Ablauf dabei. Immer muß erst eine Adresse auf den Adreßleitungen ausgesandt werden und dann müssen entsprechend der gewünschten Aktion die Steuersignale bereitgestellt werden, wobei auch das Timing dabei weitgehend natürlich ist. Denn zum Beispiel

kann ein Speicherbaustein erst einige Zeit nach seiner Anwahl Daten freigeben oder empfangen. Gleiches gilt auch für Peripheriebausteine. Das Bussystem des NDR-Computers ist nun so konstruiert, daß es besonders einfach ist und daß es die Peripherie und Speicherkarten nach Art des Z80 (mit einigen Erweiterungen für einen Ausbau mit modernen 8-/16-Bit-Prozessoren) bedienen kann. Der Z80-Prozessor ist heute einer der bewährtesten Computerchips, den die Industrie massenweise einsetzt. Die Signale anderer Prozessortypen lassen sich meist mit nur wenig Zusatzlogik an diesen Bus anpassen.

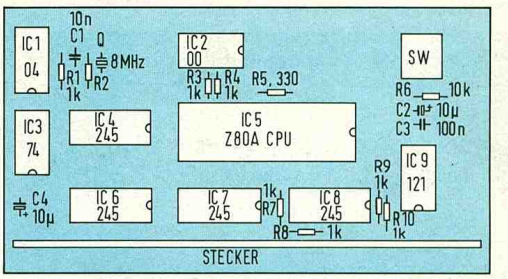


Bild 3. Der Bestückungsplan der Vollausbau-CPU

Die Vollausbau-CPU

Wenn man auf der SBC2-Platine ein Programm schreiben würde, das sehr hohe Adressen, etwa über A000h, ansprechen wollte, dann würde unter diesen Adressen wieder etwas gelesen, was auch schon unten im Adreßraum steht, denn die höherwertigen Adreßleitungen werden einfach nicht zur Speicheranwahl ausgewertet. Der Prozessor kann also in solchen Fällen Adreßbereiche doppelt sehen. Das darf natürlich keinesfalls in Systemen geschehen, wo jede Adresse auch mit Speicherzellen ausgerüstet ist. Deshalb muß eine neue CPU-Karte eingesetzt werden. Deren Schaltplan zeigt *Bild 2*.

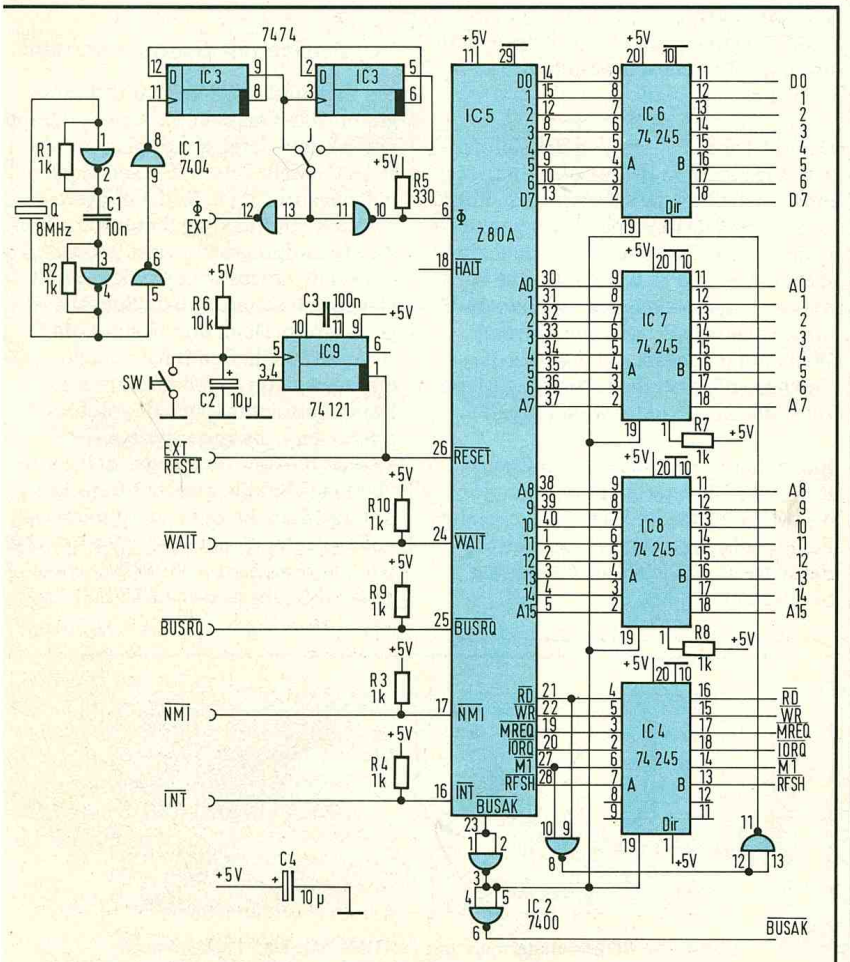


Bild 2. Die Vollausbau-CPU mit Z80, eine in sich geschlossene Baugruppe

Im Grunde sind dort die Z80-Signale nur noch über geeignete Verstärker-ICs geleitet, um dann auf den Bus zu gelangen. Auf der Platine ist neben der CPU, der Reset-Logik und der Taktlogik nichts weiter vorgesehen. Das ergibt einen klaren Funktions-Block, eben die CPU. Die Taktlogik ist etwas raffinierter aufgebaut als bei der SBC2-Platine, weil an die Präzision und die Belastbarkeit der Signale jetzt erhöhte Anforderungen gestellt werden müssen. Ein Oszillator mit 8-MHz-Quarz erzeugt einen Takt, der von IC 3 halbiert wird und dann über Verstärker sowohl an die CPU als auch an den Bus weitergegeben wird. Die Reset-Logik ist genauso wie bei der SBC2-Platine aufgebaut. Ein Monoflop (IC 9) erzeugt bei Betätigung des Schalters SW einen präzisen Impuls, der die CPU zurücksetzt, ohne den von der CPU durchgeführten Refresh dynamischer Speicherbaugruppen zu lange zu blockieren. Wiederum ist das Signal an den Bus geführt, um auch andere Karten rücksetzen zu können.

Die eigentliche Bussteuerlogik wird von den ICs 2, 4, 6, 7, 8 gebildet.

Bus-Puffer sind notwendig

Solche bidirektionalen Bus-Puffer, wie die ICs 4, 6, 7, 8 im Slang genannt werden, findet man oft in größeren Mikroprozessorsystemen. Sie werden benötigt, weil viele der hochintegrierten ICs relativ „schwache" Ausgänge besitzen. Es ist nämlich so, daß jedes an einer Signalleitung mit einem Eingang angeschlossene IC diese Signalleitung belastet, weil bei Schaltvorgängen und auch im statischen Zustand je nach Lage Strom in das Eingangsbein des angeschlossenen ICs hinein- oder aus diesem Bein herausfließt.

Und das leider immer mit der Tendenz, die Signalspannung zu schwächen. Die Last, die solch ein IC-Eingang für eine Signalleitung bedeutet, ist je nach IC-

Tabelle 1. Die Stückliste für die Vollausbau-CPU mit Z80

Stück	Bezeichnung	Bauelement
1	IC1	74 LS 04
1	I2	74 LS 00
1	IC3	74 LS 74
4	IC4, IC6, IC7, IC8	74 LS 245
1	IC5	Z80-A-CPU
1	I9	74 LS 121
4	SO14	14polige IC-Fassung
4	SO20	20polige IC-Fassung
1	SO40	40polige IC-Fassung
8	R1, 2, 3, 4, 7, 8, 9, 10	1 kΩ
1	R5	330 Ω
1	R6	100 kΩ
1	C1	10 nF
2	C2, C4	10 µF
1	C3	100 nF
1	T1	Taster
1	Q1	Quarz 8 MHz
1	St1 (Stecker)	18- und 36polige Steckerleiste
1	Platine mit Lötstoplack	

Typ unterschiedlich. Deshalb kann ein Z80-Baustein mit seinen Ausgängen nur Signalleitungen betreiben, an welchen nicht zu viele Eingänge anderer ICs angeschlossen sind. Beispielsweise darf nicht mehr als ein Standard-TTL-IC-Eingangsbein angeschlossen sein. Solche Bausteine sind eine besonders hohe Belastung. Man kann an einen Z80-Ausgang bis zu fünf TTL-LS-Baustein-Eingänge anschließen, weil diese Bausteine sich von den Standard-TTL-Bausteinen unter anderem dadurch unterscheiden, daß sie Signalleitungen nur wenig belasten. Leider sind auch fünf Eingänge an einer Z80-Ausgangsleitung oft zur Weiterverarbeitung der Signale in großen Systemen nicht ausreichend. Setzt man zur Signalverstärkung 74 LS 245-Bausteine ein, dann kann man bis zu 60 andere TTL-LS-Bausteine darüber mit Signalen versorgen. Das ist auch für Systeme mit Bussteckplätzen (wie beim NDR-Computer) ausreichend, bei welchen man ja nicht immer vorhersehen kann, wie viele ICs ein Signal belasten. Übrigens, die Anzahl der TTL-Lasten, die ein Baustein treiben kann, nennt man sein Fan Out.
Die Bustreiber sind nicht nur Verstärker, sondern auch Tri-State-Elemente, das heißt, sie können in ihrem Inneren alle Ausgänge auf „hochohmig" schalten und so die Z80-CPU (IC 5) vollständig vom Bus trennen. Mit dem Signal BUSRQ, das von außen gegeben werden muß, kann das erreicht werden. Der Prozessor gibt dann ein Signal BUSAK aus, wenn er den Bus nicht mehr benötigt.

Die LS 245-Bausteine werden über den Tri-State-Eingang (Pin 19) in den Tri-State-Zustand geschaltet. Es liegen dann nur noch Reset und der Takt direkt auf dem Bus.

Damit ist es mit speziellen Bausteinen, z. B. sogenannten DMA-Controllern, möglich, auf den Bus zuzugreifen, um so z. B. schnell an den Speicher heranzukommen. Der Eingang $\overline{\text{WAIT}}$ dient zum Anhalten der CPU, wenn z. B. eine Peripherieeinheit zu langsam ist. Dann muß diese bei einem Zugriff den Eingang $\overline{\text{WAIT}}$ auf 0 V legen, so lange, bis die Daten korrekt verarbeitet wurden. Dann kann die CPU wieder weiterarbeiten.

Interrupts können mit den Eingängen $\overline{\text{INT}}$ und $\overline{\text{NMI}}$ verarbeitet werden. Doch das ist nur etwas für Spezialisten (siehe Folge „Schrecksekunde" aus dem Sonderheft 1 und „Tick-Tack" in diesem Sonderheft).

Aufbau und Test

Der Aufbau der CPU-Platine ist besonders einfach, weil doch recht wenig ICs auf der Platine sind. Wie immer setze man zuerst die passiven Bauelemente ein, dann die IC-Sockel und erst dann die ICs selbst. Auf die Lage von Pin 1 achten! *Tabelle 1* zeigt die Stückliste, *Bild 3* die Lage der Bauelemente auf der Platine.

Die Vollausbau-CPU läßt sich zunächst nur grob vortesten, dazu wird der Takt an Pin 6 der CPU gemessen, er muß 4 MHz (normalerweise) betragen. An Pin 26 muß bei Betätigen der Reset-Taste ein kurzer Reset-Puls erscheinen. Die restliche Funktion kann man nur zusammen mit einer Speicherbaugruppe testen.

Dafür gibt es zwei Möglichkeiten. Entweder man will gleich mit CP/M arbeiten oder man will zunächst das Grundprogramm mit der Vollausbau-CPU betreiben.

Der Betrieb mit Grundprogramm

Für den Start mit dem Z80-Grundprogramm kann man die ROA64-Baugruppe verwenden, deren Schaltplan im Abschnitt „Mehr Bits" in diesem Sonderheft abgedruckt ist. *Bild 4* zeigt eine mögliche Bestückung dieser Karte. Das Z80-Grundprogramm wird in den linken Sockel (0) eingesteckt. Ein RAM-Baustein mit 8 KByte muß im Sockel 4 eingesetzt sein. Denn dort beginnt die Adresse 8000h. Damit hat man die Speicherzellen von 8000h bis 9FFFh zur freien Verfügung. Wem das nicht ausreicht, der kann mit weiteren RAM-Bausteinen die Karte ausbauen; z. B. von 8000h bis FFFFh, also 32 KByte. Die unteren Speicherbereiche können ggf. auch genutzt werden, jedoch sind sie für weitere Sprachen in EPROMs vorgesehen. So gibt es das Z80-GOSI im

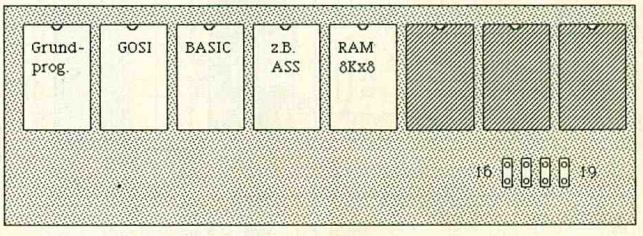


Bild 4. Die ROA64-Baugruppe mit EPROMs und RAM, schematisch dargestellt

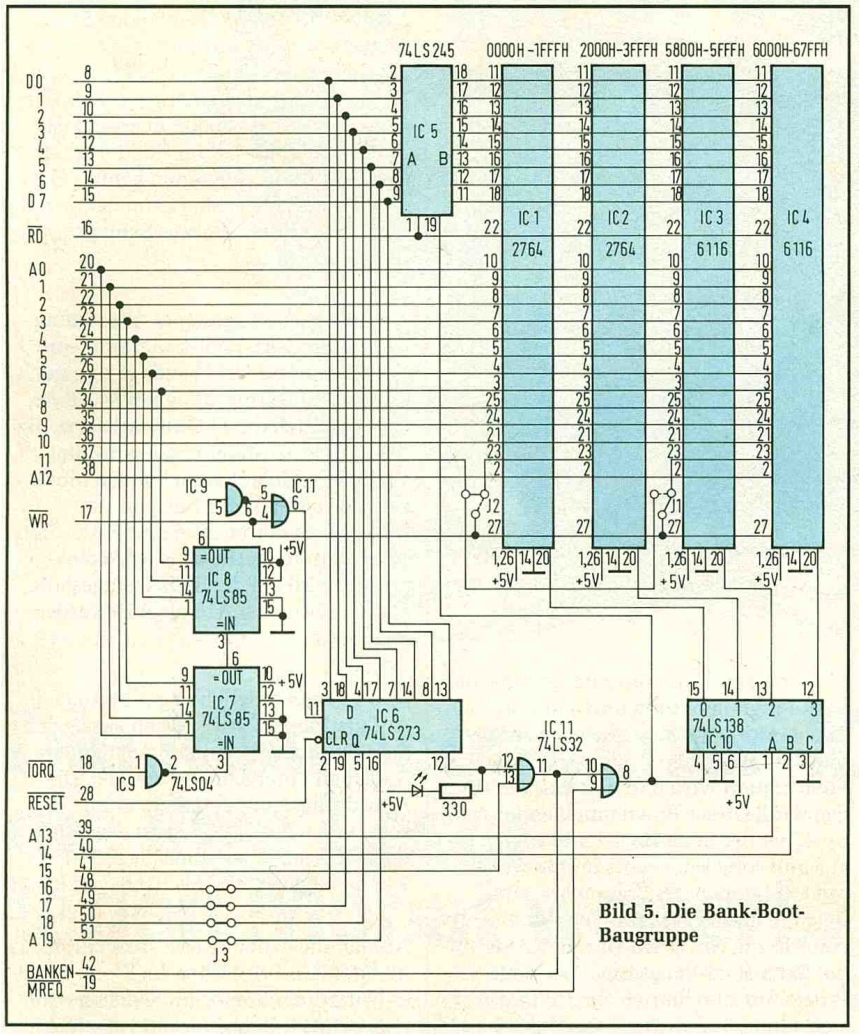


Bild 5. Die Bank-Boot-Baugruppe

8-KByte-EPROM ab Adresse 2000h (Sokkel 1) und das 8-KByte-Basic für den Sockel 2, das auf Adresse 4000h gestartet wird. Der Sockel 3 ist dann z. B. für einen Line-by-Line-Assembler gedacht, den es als Ergänzung zum Grundprogramm gibt. Mit dieser Konfiguration besitzt man schon ein sehr komfortables Computersystem, bei dem mehrere Sprachen direkt verfügbar sind. Die Brücken für die Adreßeinstellung A16 bis A19 bleiben in dieser Konfiguration offen, das heißt, es werden nur die Adressen von 0 bis FFFF angesprochen, denn die Vollausbau-CPU bedient die Adreßleitungen A16 bis A19 nicht. Sie sind offen und liegen daher auf einem Pegel logisch 1.

Die CP/M-Konfiguration

Wer noch mehr Speicher haben will, der benötigt die Bank-Boot-Karte. Damit ist der Z80 in der Lage, einen Adreßraum von 1 MByte (1 Megabyte = 1024 KByte) zu bedienen.

Ferner kann sie noch eine andere Aufgabe erledigen, die für CP/M Voraussetzung ist: den „Boot". CP/M selbst benötigt nämlich einen durchgehenden RAM-Bereich von Adresse 0 bis FFFF. Ab 0 muß aber nach einem Reset ein Programm stehen, sonst weiß der Z80 nicht, was er tun soll. Die Boot-Karte ist nun in der Lage, wahlweise in den unteren Speicherbereich RAM oder ROM einzublenden. *Bild 5* zeigt die Schaltung, *Tabelle 2* die Stückliste und *Bild 6* den Bestückungsplan.

IC 5 ist ein Bustreiber. Er dient der Bustrennung und Erhöhung des Fan Outs. IC 1 und IC 2 sind 8-KByte-EPROMs und IC 3 und IC 4 sind RAMs mit je 2 KByte (bzw. 8 KByte), die als eigener freier Speicher von manchen System-EPROMs benötigt werden. Mit den Brücken J1 und J2 kann man die Bausteintypen einstellen; auf dem Layout sind jedoch die im Schaltplan angegebenen Positionen schon vorverdrahtet.

IC 7 und IC 8 übernehmen die Decodierung des Ports. Die Port-Adresse der Baugruppe liegt fest auf C8h. Mit IC 10 wird der Speicher-Bereich von 0 bis 7FFF decodiert. Dieser Bereich ist nur dann aktiviert, wenn an Pin 12 des IC 6 (74 LS 273) ein 0-Signal ausgegeben wird. Wenn dieses Bit auf 1 geht, wird die Baugruppe abgeschaltet. Mit den Bits 0 bis 3, die am Baustein IC 6 liegen, wird der Bankbereich angegeben. Dazu sind dessen Ausgänge direkt mit den Adreßleitungen A16 bis A19 verbunden.

Bei der Brücke J3 kann man diese Leitungen aber auch unterbrechen, wenn man die Baugruppe zum Beispiel mit dem 68008 verwenden will, der die Adreßleitungen A16 bis A19 selbst treiben kann.

IC 6 wird durch IC 7 und IC 8 selektiert, wenn die I/O-Adresse C8h anliegt. *Bild 7* zeigt nochmals die genaue Bitbelegung des Ports. Bit 7 kann also den Speicherbereich der Bank-Boot-Karte ausblenden und Bit 0 bis 3 geben eine zusätzliche Bankadresse an.

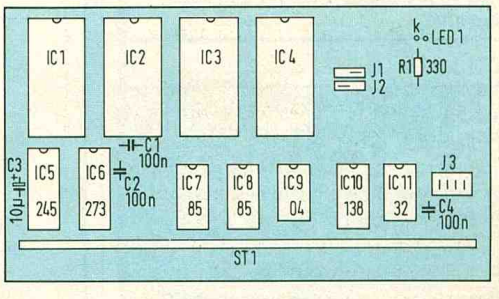


Bild 6. Der Bestückungsplan der Bank-Boot-Baugruppe

Tabelle 2. Die Stückliste zur Bank-Boot-Karte

Stück	Bezeichnung	Bauelement
1	IC5	74 LS 245
1	IC6	74 LS 273
2	IC7, IC8	74 LS 85
1	IC9	74 LS 04
1	IC10	74 LS 138
1	IC11	74 LS 32
4	SO28	28polige IC-Fassung
2	SO20	20polige IC-Fassung
3	SO16	16polige IC-Fassung
2	SO14	14polige IC-Fassung
3	C1, C2, C4	100 nF
1	C3	10 µF (Elko)
1	R1	330 Ω
1	LED1	Leuchtdiode
1	ST1 (Stecker)	18- und 36polige Steckerleiste
1	Platine mit Lötstoplack	

Die Leitung BANKEN, die an den Bus geführt ist, dient dazu, andere Speicherbaugruppen ein- oder auszublenden. Diese Leitung liegt normalerweise auf 1, wenn alle Speicher aktiviert werden. Die Bank-Boot-Karte legt das Signal immer dann auf 0, wenn eine Adresse kleiner 8000h, also 0 bis 7FFF anliegt UND das Bit 7 am Port C8 auf 0 liegt. In dieser Situation blendet sich die Karte in alle Bankbereiche von 0 bis 7FFF ein. Bei einem Reset wird das IC6 über den CLR-Eingang gelöscht und alle Ausgänge führen 0. Damit wird die Bank 0 angewählt und gleichzeitig blendet sich die Baugruppe im Adreßbereich 0 bis 7FFF ein.

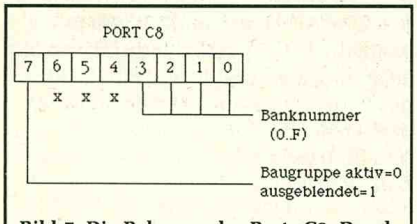


Bild 7. Die Belegung des Ports C8. Durch Einschreiben von Werten in diesen Port wird vom Z80 aus 1 MByte Speicher verwaltet

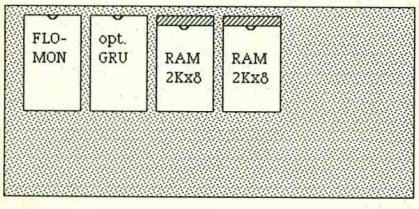


Bild 8. EPROMs und RAMs auf der Bank-Boot-Baugruppe

Boot

Ein Programm auf dieser Baugruppe, das Boot-Programm, sorgt nun dafür, daß ein Teil des EPROM-Bereichs in den allgemein zugänglichen RAM-Bereich ab F000 kopiert wird und dort angesprungen wird. Dieses Programm blendet den Speicher der Bank-Boot-Karte wieder aus und sorgt seinerseits für die weiteren Funktionen des Computers, zum Beispiel unter CP/M. *Bild 8* zeigt die Anordnung von EPROMs und RAMs auf der Bank-Boot-Baugruppe. Das neue Programm zum Betrieb von CP/M nennt sich Flomon und sitzt ganz links. Es benötigt zwei RAM-Bausteine zu je 2 KByte auf den rechten beiden Steckplätzen. Wahlweise kann man auf dem verbleibenden zweiten Steckplatz von links ein Grundprogramm stecken, das dazu auf Adresse 2000h übersetzt ist. Beide EPROMs sind im Handel erhältlich. Das Grundprogramm kennen Sie von der SBC2 her. Man kann damit viele Tests ausführen, wenn es Schwierigkeiten geben sollte.

Bild 9 zeigt die gesamte Anordnung. Neben der Bank-Boot-Baugruppe und der Vollausbau-CPU benötigt man noch eine Speicherbaugruppe mit 64 KByte, und natürlich die FLO2-Baugruppe, die die Floppy-Ansteuerung übernimmt. Weitere Speicherkarten können hinzugefügt werden, zum Beispiel um CP/M 3.0 zu fahren oder eine RAM-Floppy zu betreiben. Die erste RAM-Baugruppe wird auf die Bank 0 eingestellt, also alle Brücken A16 bis A19 werden eingesetzt.

Bild 10 zeigt mögliche Erweiterungen. Auf der Bank E0000 können zusätzlich das Grundprogramm und/oder andere Sprachen untergebracht werden. Die Anordnung kann dann wie in Bild 4 sein. Von 0 bis DFFFF kann man RAM unterbringen, oder von 0 bis FFFFF, wenn man auf EPROM-Plätze verzichtet.

Auf der Bank-Boot-Karte sitzen Flomon und der RAM-Speicher. Im Flomon gibt es Unterprogramme, mit welchen man Daten in 128-Byte-Blöcken von einer Bank zur anderen verschieben kann.

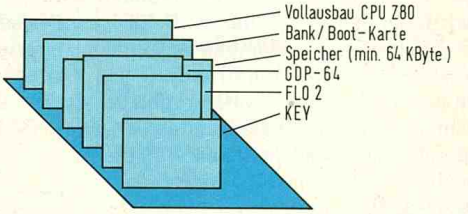


Bild 9. CP/M-80-Konfiguration

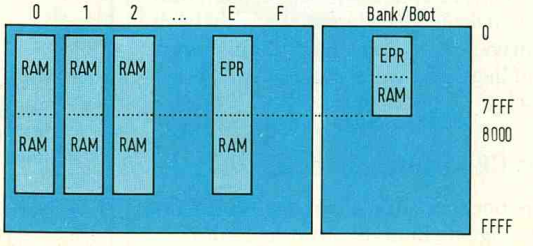


Bild 10. Alle Bänke in einer Übersicht

Die Mikrocomputer-Zeitschrift, die ihre Leser jeden Monat weiterbringt.

Ob Sie die Mikrocomputerei als Freizeitvergnügen oder aus beruflicher Notwendigkeit betreiben: Mit mc dringen Sie jeden Monat tiefer ein in die Mikrocomputertechnik. Und einsteigen können Sie jederzeit...

mc testet Hardware und prüft Programme...

mc informiert Sie umfassend über Computer und Peripherie, über Programmiersprachen und Betriebssysteme...

mc liefert Software-Ideen und regt an zum Selberbauen. Sie finden Applikationen vom einfachen Interface bis zum kompletten Selbstbausystem...

mc gibt Ihnen viel, wenn Sie mc mit ein wenig technischem Verständnis entgegenkommen...

mc macht ihre Leser Schritt für Schritt zu Mikrocomputer-Profis. Interessiert? Dann machen Sie doch einfach von unserem nebenstehenden Kennenlern-Angebot Gebrauch...

Die Mikrocomputer-Zeitschrift

Kennenlern-Angebot

Ich möchte die mc unverbindlich kennenlernen. Schicken Sie mir die beiden neuesten Ausgaben kostenlos. Informiere ich Sie danach nicht anders, abonniere ich die mc ab _____ zum Jahrespreis von DM 66,– (im Ausland DM 72,–) inkl. Porto.

Name _____

Beruf _____

Straße _____

PLZ/Ort _____

Datum/Unterschrift _____

Die Kündigung ist jeweils 8 Wochen vor Ablauf des Abonnements möglich. Wichtig: Nach Erhalt des 2. kostenlosen Heftes kann ich innerhalb von 10 Tagen durch einfache schriftliche Mitteilung an den Verlag von einem Abonnement Abstand nehmen.

Bitte hier Ihre zweite Unterschrift _____ SOS2

Bitte ausschneiden und einsenden an:

Franzis' Franzis Verlag, Postfach 37 01 20, 8000 München 37

Rolf-Dieter Klein

Die Floppy-Karte

FLO2 für den NDR-Klein-Computer

Hier folgt die Floppy-Steuer-Karte für den NDR-Klein-Computer. Jeder, der schon eine der Systemkarten erfolgreich aufgebaut hat, kann auch diese Karte nachbauen. Damit besitzt er dann die Schlüsselkarte, die seinen Computer zu einem professionellen Gerät macht. Übrigens kann die Schaltung auch am mc-CP/M-Computer arbeiten, allerdings nur mit einem neuen EPROM und entweder über eine Adapterkarte oder über eine eigene Layout-Version.

mc hat schon in Heft 7/1984 über die Technik berichtet, mit der Daten auf Disketten geschrieben und auch wieder davon gelesen werden können. Das Thema ist deshalb nicht leicht zu behandeln, weil es dabei die verschiedensten Formate und Verfahren gibt. Unser Format wird nun, anders als beim mc-CP/M-Computer, 80-Spur-Laufwerke in 5¼-Zoll-Technik ansprechen und mit der MFM-Technik arbeiten. Weshalb diese Umstellung? Die technische Entwicklung ist seit September 1982, da erschien der mc-CP/M-Computer, so weit fortgeschritten, daß das damals verwendete Format nicht mehr zeitgemäß ist. Wir haben uns bemüht, jetzt einen zukunftssicheren Standard festzulegen. Da bei dieser Festlegung die Väter aller mc-Computer mitbestimmt haben, die für ihre Kinder nur das Beste akzeptieren, sind wir sicher, das Richtige gefunden zu haben, das jahrelang Bestand haben wird. Das mc-Format wird von allen mc-XXXXX-Computern, die nicht den 6502 als CPU besitzen, physikalisch gelesen werden können. Wir hoffen damit, daß mc-Leser bald problemlos zumindest Daten- und Text-Dateien austauschen können. Und daß Sie erkennen, daß mc-Systeme nicht ohne Überlegung entworfen werden.

FLO2 und seine Chips

Auf jeder Floppy-Karte sitzt mit dem Controller-Chip ein Baustein, der selbst ein sehr intelligenter Prozessor ist. Mit einigen wenigen Befehlen kann man komplexe Suchaktionen und vollständige Datenübertragungen auslösen, die vom Controller selbständig ausgeführt werden. Das hier verwendete IC 1797 (Lieferant zum Beispiel Siemens) besitzt unter anderem den Befehl Lies Spur, mit dem eine vollständige Diskettenspur komplett gelesen werden kann. Der Controller-Chip erledigt also die logischen und verwaltungstechnischen Aufgaben auf der FLO2-Platine. Er findet die richtige Spur, er meldet die Ergebnisse eines Suchprozesses an die CPU und er liefert auch die Daten dorthin ab. Dabei wird die Interrupt-Technik benutzt, um der CPU zu signalisieren, daß ein zu bearbeitendes Ergebnis vorliegt. Der zweite wichtige Baustein auf der Platine ist der Datenseparator-Baustein 9229, der die Schreib- und Lesesignale vom und zum 1797 so aufbereitet, daß sich die Elektronik des Laufwerkes und der Prozessor 1797 richtig verstehen. Im Prinzip geschieht dabei Folgendes (in mc 1984, Heft 7 ist das genau geschildert): Erstens wird das vom Floppy-Controller schon korrekt ausgegebene Schreibsignal zur Aufzeichnung in manchen Fällen noch etwas elektronisch zurechtgebogen, damit es später fehlerfrei wieder gelesen werden kann. Diese Vorbehandlung nennt man Präkompensation. Sie ist bei 8-Zoll-Laufwerken für die inneren Spuren notwendig (bei anderen Laufwerken meist überflüssig); zweitens nimmt der Baustein 9229 beim Lesen die Trennung zwischen den Synchronsignalen und den eigentlichen Datensignalen vor, die ja bei der Aufzeichnung zusammengemischt wurden. Aus diesem Grund heißt der Baustein 9229 auch Datenseparator. Daß er dabei Toleranzen ausregelt, die durch ungleichmäßigen Lauf der Magnetscheibe entstehen, sei hier nicht verschwiegen.
Weil der Separator-Baustein intern so raffiniert aufgebaut ist, kann man die Platine jetzt ohne Abgleich-Probleme aufbauen. Ohne weiteres kann man jetzt bei 8-Zoll-Laufwerken Double Density fahren. Man kann also jetzt jeden Laufwerktyp, sei es 3-Zoll, 3½-Zoll, 5¼-Zoll oder 8-Zoll, mit einfacher oder doppelter Dichte bedienen. Ebenso können Laufwerke mit nur einem Schreibkopf, Doppelkopf-Laufwerke und Laufwerke mit 80 Spuren verwendet werden. Auch gemischter Betrieb ist möglich.

Die Signale für FLO2

Über den bidirektionalen Bustreiber (IC 10, 74LS245) ist der Datenbus des Computers mit dem Controller-Baustein verbunden (*Bild 1*). Ein Widerstandsnetz mit acht Widerständen von je 3,3 kΩ sorgt dafür, daß keine Störungen auf der Versorgungsleitung auftreten, wenn der Buszustand am Controller von Tri State auf Aktiv wechselt. Solche Störungen können von 74LS245-ICs neuerer Generation verursacht werden, da diese bei einem Wechsel von Offen nach Aktiv schlagartig an den B-Eingängen viel Strom ziehen. Durch die Pull-up-Widerstände wird verhindert, daß die Spannung an den B-Eingängen dabei unter 2,5 V absinkt, was Signalverfälschungen verhindert.
Am Vergleicher (IC 13, 74LS85) kann die Adresse der Baugruppe mit Brücken eingestellt werden. Sie lautet 0C0h, wenn die Brücken gegenüber von A4 und A5 auf Masse geschaltet werden und die Brücken gegenüber von A7 und A6 offen bleiben. Auf dem Bestückungsplan ist die Lage der Brücken eingezeichnet. Das IC gibt jetzt bei OUT genau dann 1-Pegel aus, wenn auf den Adreßleitungen die Signalkombination A4 = 0, A5 = 0, A6 = 1 und A7 = 1 auftritt.
Von dem Decoder (IC 14, 74LS138) werden zwei Adreßbereiche angesprochen. Von 0C0h bis 0C3h wird der Floppy-Controllerbaustein (IC 4) aktiviert und von 0C4h bis 0C7h werden beim Schreiben das IC 12 und beim Lesen das IC 11 angewählt. Dabei ist jedesmal aber nur die Adresse 0C4h gemeint, denn die anderen Adressen sind einfach nur nicht eindeutig decodiert und wählen ebenfalls die genannten Bausteine an.

Das IC 12 (74LS273) speichert die Laufwerksnummer und gibt sie an den Stekker zum Laufwerk. Außerdem werden Laufwerktyp (Mini oder Maxi) sowie Schreib- und Lesedichte (single, double) und Seitenauswahl-Signal vom Prozessor dort erwartet.
Sowohl die Adressierung, damit die Karte sich angesprochen fühlt, als auch die Anwahl und Einstellung der einzelnen Laufwerke muß also die CPU durchführen.
Bild 3 zeigt die Bedeutung der Bits in IC 12. Die Bits 0 und 3 dienen zum Einstellen des Laufwerks. Wenn ein Laufwerk angesprochen werden soll, so wird einfach das entsprechende Bit auf 1 gesetzt. Dabei ist die Laufwerkscodierung also nicht dual durchgeführt, sondern durch Anwahl von genau einer von 4 Leitungen. Wenn man über Laufwerke verfügt, die intern eine duale Anwahl verkraften können, dann könnte man bis zu 16 Laufwerke anschließen. Solche Laufwerke sind selten.
Mit Bit 4 wird bestimmt, ob die Aufzeichnung in einfacher Dichte (FM, Bit 4 = 1) oder doppelter Dichte (MFM, Bit 4 = 0) erfolgen soll.
Mit Bit 5 wird bestimmt, ob 5¼-Zoll- oder 8-Zoll-Laufwerke verwendet werden sollen. Wird das Bit 5 auf 1 gesetzt, so ist der Takt für den Floppy-Controller auf die Hälfte herabgesetzt. Das ist die richtige Einstellung für die 5¼-Zoll-Mini-Laufwerke. Auch die meisten kleineren Laufwerke, die Mikrolaufwerke, kommen damit zurecht. Die Aufzeichnungsfrequenz bei den großen 8-Zoll-Laufwerken ist doppelt so groß, was durch Nullsetzen von Bit 5 auch auf der Controller-Platine eingestellt werden kann.

Mit Bit 6 kann man den Motor der Laufwerke schalten, wenn die Brücke bei ST3 so eingebaut ist, wie sie im Bestükkungsplan eingezeichnet ist.

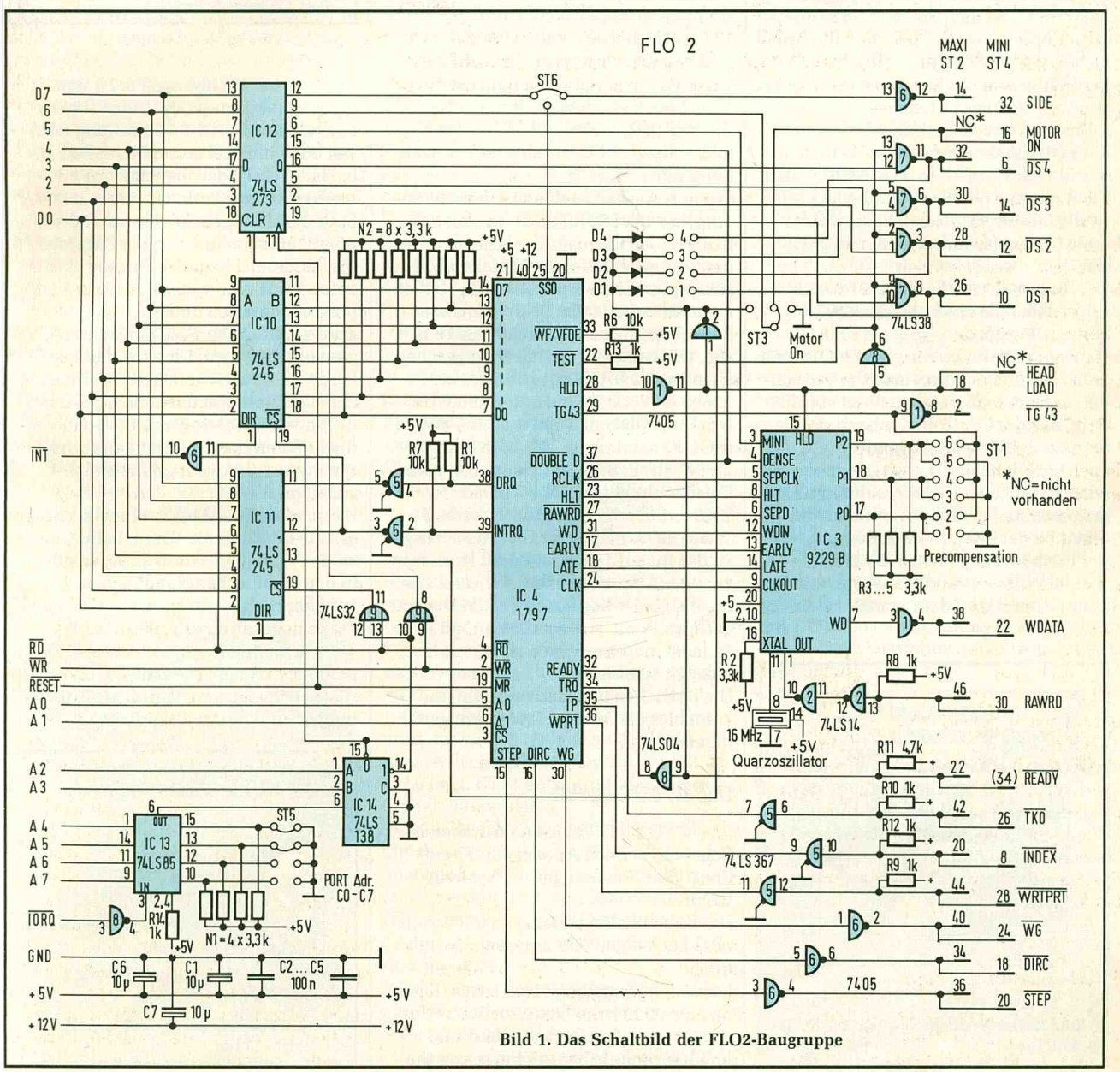


Bild 1. Das Schaltbild der FLO2-Baugruppe

Wenn Bit 6 auf 0 steht, so ist der Motor eingeschaltet, wenn auf 1, so ist er ausgeschaltet. Wenn man das Bit zum Schalten verwenden will, so muß man darauf achten, daß entweder das Laufwerk einen READY-Ausgang besitzt oder daß man nach dem Einschalten per Warteschleife so lange wartet, bis der Motor seine Nenndrehzahl erreicht hat. Andernfalls können insbesondere beim Schreiben Fehler auftreten, die defekte Sektoren verursachen. Beim Lesen ist das anders. Da wird ein Sektor vom Controller automatisch immer wieder gelesen, wenn sich bei der Fehlerüberprüfung herausstellt, daß er von der Diskette fehlerhaft abgetastet wurde.
Mit Bit 7 kann man die Floppy-Seite auswählen, wenn man Laufwerke mit zwei Köpfen besitzt. Dazu muß die Brükke St6, wie im Schaltbild eingezeichnet, verdrahtet sein. Im Layout ist die Brücke bereits fest eingebaut, daher muß man normalerweise keine Veränderung an der Leiterplatte vornehmen. Wenn man das Bit 7 auf 1 setzt, so wird Seite 1, also die Rückseite der Floppy-Scheibe (das ist die mit Beschriftung!) ausgewählt. Seite 0 ist angewählt, wenn man das Bit 7 auf den Wert 0 setzt.
Der Floppy-Controller FD1797 besitzt selbst ebenfalls einen Ausgang für die Seitenauswahl, den Ausgang SS0. Die Verwendung dieses Ausgangs hat jedoch große Nachteile. Wenn man ihn verwendet, so kann man nämlich nicht verhindern, daß der Controller anhand des Datenrecords automatisch kontrolliert, ob auch wirklich Seite 1 angesprochen wird. Es gibt aber Disketten mit Programmen im Handel, bei welchen das Seitenbit der Sektorkennzeichnung bei der Formatierung nicht auf 1 gesetzt wurde. Will man solche Disketten lesen können, muß man dem Controller einen Befehl geben können, nach dem er glaubt auf Seite 0 zu lesen und zu prüfen während Seite 1 angewählt ist. Das gelingt nur, wenn man SS0 selbst erzeugt, wie zum Beispiel auf FLO2.

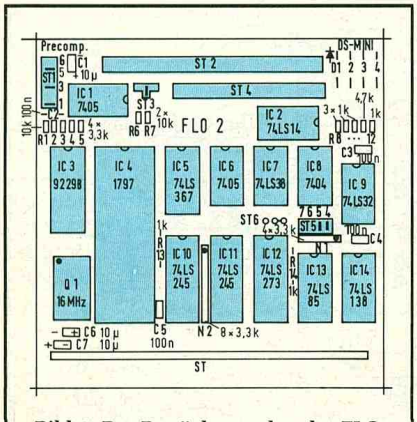


Bild 2. Der Bestückungsplan der FLO2-Baugruppe

So wird gelesen

Bild 4 zeigt die Bedeutung der Bits unter Adresse 0C4h beim Lesen. Bit 7 ist das DRQ-Bit des Controllers. Leider kann man es nicht ohne Nachteile vom Controller direkt abnehmen. Daher mußte es über das IC 11 geführt werden. Wenn dieses Bit auf 1 liegt, so will der Controller ein Byte holen oder hat ein Byte bereitgestellt. DRQ wurde auf Bit 7 des Datenbusses gelegt, damit man eine schnelle Abfrage des Bits (durch einen RLCA-Befehl beim Z80, oder ROL.B-Befehl beim 68000/8) erreichen kann. Das Bit landet nämlich schnell im Carry-Flag. Man kann danach durch einen Sprung „JP C," bzw. „JP NC," oder „BCS" bzw. „BCC" rasch eine Verzweigung durchführen.
Genauso verhält es sich mit dem Bit 6, dort liegt der INTRQ-Ausgang des Controllers. Das Bit wird auf 1 gesetzt, wenn der Controller eine Unterbrechung des Hauptprozessors erreichen will, weil er etwas mitzuteilen hat. Nun kann man zum einen einen echten Interrupt auslösen, wie es bei der Z80-CPU vorgesehen ist, oder das Bit abfragen, wie es bei unseren 68000/8-Routinen getan wird. Das Bit landet nach einem RLCA-Befehl (ROL.B) nämlich im Vorzeichenbit. Mit „JP M" (JP P, BMI, BPL) kann dann eine Entscheidung getroffen werden.
Bit 5 enthält die Headload-Information: Wenn der Kopf auf der Diskette aufliegt, ist das Bit auf 1 gesetzt. Damit kann man vor einem Schreib- und Lesebefehl abfragen, ob der Kopf noch auf der Diskette aufliegt. Wenn ja, braucht man ihn nicht zu laden und kann dadurch 15 ms Kopfladezeit sparen.
Die restlichen Bits sind unbelegt und können später einmal Erweiterungen dienen.

Das IC FDC-9229B

Das IC FDC-9229B ist ein universeller Baustein, der den Aufbau von Floppy-Controller-Schaltungen stark vereinfacht.
Das IC beinhaltet neben dem Datenseparator auch eine Präkompensationsschaltung.
Der Datenseparator ist sehr ausgeklügelt. So ist zum Beispiel kein Abgleich erforderlich, was den Nachbau der FLO2 erheblich vereinfacht. Der Präkompensator besitzt drei Eingänge P0, P1 und P2 mit welchen man seine Zeiten flexibel einstellen kann. In der Schaltung wird das mit den Brücken bei ST1 getan. Dabei ist wichtig, daß manche Floppy-Laufwerke eine Präkompensation erst ab Spur 43 benötigen. Es gibt daher drei Möglichkeiten für jeden der Eingänge P0 bis P2. Einmal liegt der Eingang über einen Widerstand an +5 V. Dann ist die Präkompensation dauerhaft eingestellt. Zum Zweiten kann der Eingang auf 0 V geschaltet werden. Dann ist das betreffende Bit unwirksam. Wird der Eingang zum Dritten auf den Ausgang TG43 des Floppy-Controllers geschaltet, dann ist die Präkompensation erst dann wirksam, wenn eine Spur größer gleich 43 angefahren wird.
Für moderne 5¼-Zoll-Laufwerke genügt es, alle Eingänge auf 0 zu schalten, also keine Präkompensation zu verwenden. In den technischen Handbüchern der Laufwerke steht ein Hinweis, falls eine Präkompensation erforderlich ist.
Tabelle 1 enthält die möglichen Präkompensationszeiten. Die Zeiten sind unterschiedliche, je nachdem ob die Einstellung Mini oder Maxi vorliegt.

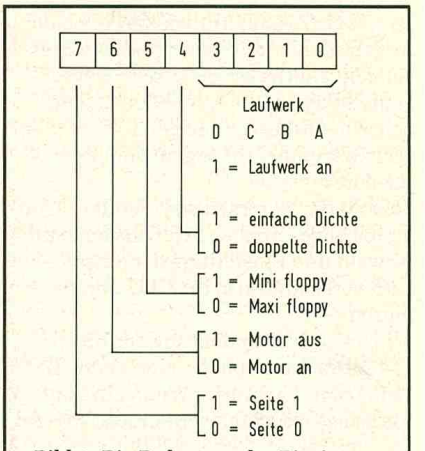


Bild 3. Die Bedeutung der Bits in Port C4 beim Schreiben

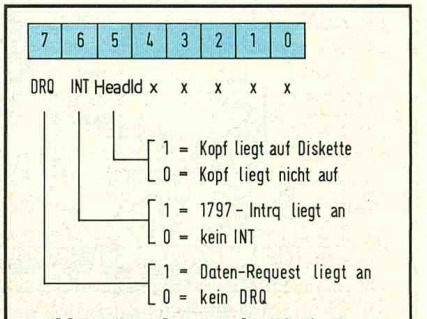


Bild 4. Die Bedeutung der Bits in Port C4 beim Lesen

Bild 5 zeigt die Einstellung für ein Maxi-Laufwerk mit mehr als 40 Spuren, das eine Präkompensation von 62,5 ns benötigt, wenn eine Spur größer gleich 43 gewählt wird. Man kann auch Präkompensationszeiten einstellen, die auf allen Spuren gleich sind, oder ab einer bestimmten Spur einen höheren Wert annehmen. Aber, wie schon gesagt, die meisten Laufwerke benötigen keine Präkompensation. Das gilt insbesondere für Minilaufwerke mit 40 Spuren. Präkompensation ist übrigens nur bei doppelter Aufzeichnungsdichte überhaupt relevant.
Bei der Präkompensation werden nahe beieinander stehende Datenbits für die Aufzeichnung noch näher zusammengedrängt. Denn bei einer Wiedergabe erscheinen sie gegeneinander verzögert, da sie vom Kopf beim Lesen „auseinandergerückt" werden. Die Präkompensationszeit gibt an, um wieviel die Bits zusammengedrängt werden. Wählt man diese Zeit zu groß, so gibt es Datenfehler, denn dann kann der Controller die Daten nicht mehr lesen. Also für die ersten Versuche, die Brücken 1, 3 und 5 bei ST1 einsetzen.

Die Steuerleitungen

Eine wichtige Funktion erfüllt das Ready-Signal, das von dem angesprochenen Laufwerk kommt. Bei allen Maxi-Laufwerken ist es vorhanden und liegt auf Pin 22 der 50poligen Steckerleiste. Bei Minilaufwerken ist es oft nicht vorhanden. Bei Laufwerken, die keinen Kopfladamagnet besitzen und die den Motor daher abschalten müssen, wenn sie nicht angesprochen werden, ist immer ein Ready-Signal vorhanden. Wenn ein Ready-Ausgang vorhanden ist, so liegt er meist auf dem Stift 34 der 34poligen Steckerleiste. Er muß dann in unserer Schaltung per Hand auf den gemeinsamen Ready-Eingang der Dioden (Anode) D1 bis D4 verdrahtet werden. Die Dioden selbst dürfen dann nicht eingesetzt werden. Wer den Schaltplan genau analysiert, stellt fest, daß sowohl $\overline{\text{READY}}$ als auch IC 1 dieselbe Datenleitung bewegen. Das geht ohne Kurzschluß, weil beide Signale von Open-Collector-Ausgängen herrühren.
Die Dioden werden für die Laufwerke benötigt, die keinen Ready-Ausgang besitzen. Wenn man gemischt arbeitet, dürfen die Dioden nur für solche Laufwerke eingesetzt werden, die keinen Ready-Ausgang besitzen (auch bei gemischtem Mini- und Maxilaufwerksbetrieb). Die Diode D1 ist dabei für das Laufwerk A (Drive select 1), die Diode D2 für das Laufwerk B usw. zuständig. Man kann insgesamt vier Laufwerke anschließen, wenn man die „eins aus vier" Codierung verwendet. Dabei kann jedes Laufwerk zusätzlich doppelseitig sein.
Bild 6 zeigt Beispiele für die Diodenbestückung. Bei binärer Codierung sind im Prinzip 15 Laufwerke möglich. Man muß dabei mit der Ready-Leitung aufpassen. Da dies aber nur ganz seltene Laufwerke betrifft, sei hier nicht weiter darauf eingegangen. Übrigens liegt Ready an der Stiftleiste auf 0, wenn das Laufwerk bereit ist.
Die anderen Steuer- und Meldesignale sind nicht so aufregend: Mit Side wird die Laufwerksseite bestimmt. Side liegt auf 0, wenn die Seite 1 ausgewählt wird, bei Seite 0 liegt es auf 1.
Motor On wird meist nur für Minilaufwerke verwendet. Es liegt auf 0, wenn der Motor an sein soll.
$\overline{\text{DS1}}$ bis $\overline{\text{DS4}}$ liegen auf 0, wenn das entsprechende Laufwerk angesprochen wird. Achtung, bei manchen Minilaufwerken gibt es kein $\overline{\text{DS4}}$. Mit Head Load wird der Laufwerkskopf an die Diskettenoberfläche gebracht, wenn das Signal auf 0 liegt. Es ist bei uns mit dem Selekt-Signal verkoppelt.
$\overline{\text{TG43}}$ liegt auf 0, wenn eine Spur größer gleich 43 angesprochen wird. Dieses Signal wird nur von älteren Maxi-Laufwerken verwendet, um den Schreibstrom zu reduzieren. Manchmal liegt der Anschluß bei dem Laufwerk auch an Stift 8 und nicht an 2.
WDATA sind Schreibdaten. RAWRQ sind die Lesedaten. $\overline{\text{TK0}}$ liegt auf 0, wenn der Kopf auf Spur 0 liegt. Es gibt ein paar Laufwerke bei welchen die Positionierung auf Spur 0 nicht funktioniert. Dort kann der Kopf auch noch auf Spur-1 fahren, was die Spur-0-Meldeschaltung durcheinanderbringen kann. Meist läßt sich das durch eine mechanische Neueinstellung beheben.
$\overline{\text{WRTPRT}}$ ist der Schreibschutzausgang. Damit kann das Laufwerk signalisieren, daß ein Schreibschutz gesetzt ist.
$\overline{\text{WG}}$ ist ein Freigabesignal für den Schreibausgang. $\overline{\text{DIRC}}$ und $\overline{\text{STEP}}$ dienen der Ansteuerung des Schrittmotors zur Spureinstellung.

Tabelle 1.
Die Präkompensationszeiten des 9229

P2	P1	P0	
Mini = 0 (also Maxi)			
0	0	0	0 ns
0	0	1	62,5 ns
0	1	0	125 ns
0	1	1	187,5 ns
1	0	0	250 ns
1	0	1	250 ns
1	1	0	312,5 ns
1	1	1	312,5 ns
Mini = 1 (also Mini)			
0	0	0	0 ns
0	0	1	125 ns
0	1	0	150 ns
0	1	1	375 ns
1	0	0	500 ns
1	0	1	500 ns
1	1	0	625 ns
1	1	1	625 ns

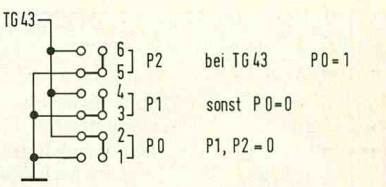


Bild 5. Eine Einstellung der Präkompensation als Beispiel

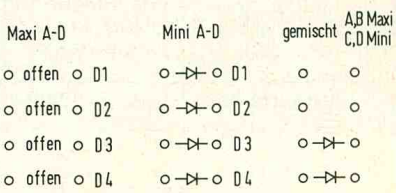


Bild 6. So werden die Dioden eingestellt

Der Floppy-Controller

Der Floppy-Controller wird über die Adressen 0C0h bis 0C3h angesprochen. *Tabelle 2* zeigt die Belegung. Die Adresse 0C0h spricht beim Schreiben das Befehlsregister an. Man kann dort Befehle an den Controller übergeben. Beim Lesen erfährt man den Status, so zum Beispiel, ob ein Lesefehler aufgetreten ist oder nicht.

Tabelle 2. Die Bedeutung der Register des Floppy-Controllers

	Lesen	Schreiben	
C0	Status	Befehl	
C1	Spur-Register	Spur-Register	
C2	Sektor-Register	Sektor-Register	1797
C3	Daten-Register	Daten-Register	
C4	Zusatz-Info	Auswahl	Zusatz

Tabelle 3. Die Befehle des 1797

Typ	Befehl	Befehl	7	6	5	4	3	2	1	0
I	Restore	Auf Spur 0	0	0	0	0	h	V	R0	R1
I	Seek	Spur suchen	0	0	0	1	h	V	R0	R1
I	Step	schreiten	0	0	1	u	h	V	R0	R1
I	Step In	schreite nach Innen	0	1	0	u	h	V	R0	R1
I	Step Out	schreite nach Außen	0	1	1	u	h	V	R0	R1
II	Read Sektor	Lies Sektor	1	0	0	m	F_2	E	F_1	0
II	Write Sektor	Schreibe Sektor	1	0	1	m	F_2	E	F_1	a_0
III	Read Address	Lies Adresse	1	1	0	0	0	E	F_1	0
III	Read Track	Lies Spur	1	1	1	0	0	E	F_1	0
III	Write Track	Schreibe Spur	1	1	1	1	0	E	F_1	0
IV	Force Interrupt	Interrupt auslösen	1	1	0	1	I_3	I_2	I_1	I_0

Das Register 0C1h ist das Spur-Register, dort steht die aktuelle Spurnummer, die von 0 bis maximal 255 reichen darf. Bei 8-Zoll-Laufwerken ist die größte Spurenzahl 76 und bei Minilaufwerken 34, 39 oder 79.
0C2h ist das Sektorregister, dort steht die aktuelle Sektornummer, auf die zugegriffen wird. Im Register 0C3h werden Parameter und Daten vom Controller an den Computer oder vom Computer an den Controller gegeben.
0C4h gehört, wie schon bekannt, nicht zum Floppy-Controller. Die *Tabelle 3* zeigt eine Liste der Befehle. Man unterscheidet vier Typen. Die erste Gruppe (I) sind Befehle für die Kopfpositionierung, die Gruppe II sind Befehle zum Lesen und Schreiben von Sektoren, die Gruppe III sind Befehle für die Kontrolle oder zum Formatieren. In der Gruppe IV sitzt nur ein Befehl für die Interruptsteuerung.
Generell wird vom Controller immer nach Ausführung eines Befehls die Leitung INT aktiviert und damit das Bit 6 am Port 0C4h gesetzt sowie INT auf 0 gesetzt. Wenn der Interrupt der CPU durch einen EI-Befehl beim Z80 (oder entsprechendes beim 68000/8) vorher freigegeben war, wird der Prozessor also zur Reaktion gezwungen. Wenn die CPU dann das Status-Register des Controllers liest, so wird das Signal INTRQ wieder gelöscht, also Bit 6 auf 0 zurückgesetzt.
Bei den Befehlen sind einige Optionen möglich. So kann man die Steprate bestimmen mit der die Kopfpositionierung durchgeführt wird. *Tabelle 4* zeigt die Umrechnungstabelle. Dann gibt es noch einige Hilfsbits, deren Bedeutung *Tabelle 5* zeigt. Wenn man das Bit mit der Bezeichnung „h" auf 1 setzt, so wird der Kopf zu Beginn des entsprechenden Befehls geladen, sonst erfolgt nur ein Update der internen Register. Bei den STEP-Befehlen muß zum Update zusätzlich das Bit U auf 1 gesetzt sein.
Wird das Bit V auf 1 gesetzt, so erfolgt ein Prüfvorgang. Der Kopf eines Records wird angelesen und der Inhalt mit der aktuellen Spurnummer, und ggf. Seitennummer verglichen. Stimmen die Werte nicht überein, erfolgt eine Fehlermeldung.

Tabelle 4. Die Stepraten

R1	R0	Maxi	Mini	Monitor
0	0	3 ms	6 ms	0
0	1	6 ms	12 ms	1
1	0	10 ms	20 ms	2
1	1	15 ms	30 ms	3

Beim Befehl „Spur suchen" wird die einzustellende Spur im Port-Register 0C3h übergeben. Step schreitet einen Schritt in die Richtung in die zuletzt geschritten wurde, Step In schreitet einen Schritt in Richtung Spur 76 (80 usw.). Step Out schreitet einen Schritt in Richtung Spur 0. Beim Schreiben oder Lesen wird zusätzlich die Sektorinformation im Register 0C2h ausgewertet. Dieser Sektor wird angewählt. Normalerweise fängt man bei Sektoren mit 1 an zu zählen, bei Spuren mit 0. Die Daten werden im Register 0C3h ausgetauscht. Beim Schreiben kündigt das DRQ-Signal die Anforderung eines neuen Wertes an, beim Lesen zeigt DRQ an, daß ein Byte vorliegt. Das DRQ-Bit wird jeweils gelöscht, wenn man ein Byte geliefert oder geholt hat. Der Controller gibt eine Fehlermeldung aus, wenn man dabei so spät reagiert, daß er nicht mehr fortlaufend mit dem Takt der Aufzeichnung auf der Floppy arbeiten kann.
Die Belegung des Statusregisters zeigt *Tabelle 6*. Sie ist abhängig vom ausgeführten Befehl und unterscheidet Typ-I- und Typ-II/III-Befehle. Mit dem Befehl „Read Adress" kann man den Startkopf eines Sektors lesen, der Auskunft über aktuelle Spur, Seite, Sektor und Sektorlänge gibt. Die Information wird in der angegebenen Reihenfolge nacheinander im Register 0C3h, also dem Datenregister mit einer DRQ-Anforderung übergeben.
Mit dem Befehl „Lies Spur" wird eine komplette Spur, beginnend beim Index-

Tabelle 5. Die Bedeutung der speziellen Bits aus Tabelle 3

h = 1	Kopf bei Start laden
h = 0	Kopf bei Start heben
v = 1	Spur prüfen durch Anlesen
v = 0	keine Prüfung
u = 1	Spur-Register auf Stand bringen
u = 0	Spur-Register belassen
m = 0	einen Record bearbeiten
m = 1	mehrere Records bearbeiten
d_0 = 0	Data-Mark FB
d_0 = 1	Deleted Data-Mark F8
F_2 = 1	Sektorlänge = Std. (IBM) 128, 256, 512, 1024
F_2 = 0	spezielle Längen = 256, 512, 1024, 128
F_1 = 0	SSO auf 0 (Seite 0 prüfen)
F_1 = 1	SSO auf 1 (Seite 1 prüfen)
I_0 = 1	Nicht Ready → Ready gibt INT
I_1 = 1	Ready → Nicht Ready gibt INT
I_2 = 1	Index Puls
I_3 = 1	sofort INT auslösen
I_3–I_0 = 0	Stop ohne Interrupt

Loch bis zum erneuten auftreten des Indexes eingelesen. Dabei werden nicht nur die Daten übertragen, sondern auch alle Zusatzinformationen und Datenlücken, also überhaupt alles, was sich auf der Spur befindet. Die Daten sind immer korrekt, da sich der Controller bei Synchrontakten immer wieder neu auf den Floppy-Takt einregelt. Dieser Befehl ist weniger zum Lesen aktueller Daten als vielmehr zu Kontrollzwecken gedacht.
Der Befehl „Write Track" schließlich dient zum Formatieren einer Diskette. Bestimmte Bytes werden dabei als Steuerinformation zur Ablage von CRC oder Synchronisationsbits interpretiert. *Tabelle 7* zeigt die Bedeutung des Bytes. *Bild 7* zeigt als Beispiel die Formatierung nach IBM mit einfacher Dichte auf 8 Zoll. *Bild 8* zeigt ein Beispiel für die Formatierung mit doppelter Dichte, auch auf 8 Zoll.
Der eigentliche Datentransfer geschieht mit den Befehlen Lese Sektor und Schreibe Sektor, nachdem die gewünschte Spurnummer und die richtige Sektornummer eingestellt wird. Dann bietet der Controller die Bytes im Sektor im Abstand von 18 µs (große Floppy DD) oder 36 µs (Mini-Floppy) an seinen Datenleitungen an oder erwartet sie dort.
Nun bleibt noch der Befehl „Force Interrupt". Damit kann man zum einen den Controller rücksetzen, zum anderen das Interruptverhalten einstellen, wie in *Bild 11*, bei den Bits I0 bis I3 dargestellt.

Zum Aufbau und zum Test

Beim Aufbau beginne man zunächst mit dem Einsetzen der IC-Fassungen. Dann werden alle passiven Bauteile, wie Widerstände und Kondensatoren eingelötet und auch die Stiftleisten.
Als nächstes löte man den Quarzoszillator Q1 ein. Dabei ist auf die Orientierung zu achten. Der Quarzoszillator besitzt an einer seiner Gehäuseseiten einen Punkt. Dieser Punkt kennzeichnet Pin 1. Er muß mit dem Punkt auf dem Bestükkungsplan übereinstimmen. Beim Quarzoszillator darf man nicht zu lange löten. Da der Baustein nur vier Anschlüsse besitzt, wäre die mechanische Stabilität nicht sehr hoch, wenn man den Oszillator mit einer Fassung montieren würde.
Nach dem Einbau des Oszillators kann man alle ICs einstecken. Der Floppy-Controller benötigt eine zusätzliche Versorgungsspannung von +12 V, die über den Bus geführt wird. Bitte das Vorhandensein dieser Spannung nochmals kontrollieren, bevor die Baugruppe in Betrieb genommen wird. *Bild 9* zeigt eine Zusammenfassung aller Brücken, wie sie auf die unterschiedlichen Stiftleisten (Ausnahme ST6) gesteckt werden müssen.
Für den Test gilt:
1. Betrieb zunächst ohne Floppy-Laufwerk. Zum Test wird entweder die SBC2-Baugruppe mit Grundprogramm allein oder die Vollausbau-CPU mit Speicher und Grundprogramm auf der Bank/Boot oder die mc-CP/M-CPU mit dem Monitorprogramm verwendet. Beim 68000/8 wird das Grundprogramm zum Test benötigt.

Tabelle 6. Die Bedeutung der Bits im Statusregister

	7	6	5	4	3	2	1	0
Befehlstyp I	1 = Nicht Ready	1 = Schreibschutz	1 = Kopf geladen	1 = Suchfehler	1 = CRC-Fehler	1 = Spur 0	1 = Index	1 = Busy
Befehlstyp II, III	1 = Nicht Ready Der Befehl wird nicht ausgef. bis Ready 0	1 = Schreibschutz	1 = Typ-Fehler 1 = Deleted Data Mark oder Schreibfehler	1 = Record nicht gefunden	1 = CRC-Fehler	1 = Datenverlust	1 = DRQ	1 = Busy

Tabelle 7. Die Formatier-Bytes

	einfache Dichte	doppelte Dichte
00–F4	Daten FM	Daten MFM
F5	–	A1* MFM, CRC löschen
F6	–	C2** MFM
F7	2 CRC-Bytes	2 CRC-Bytes
F8–FB	F8–FB mit CLK = C7, CRC löschen	F8–FB MFM
FC	FC mit CLK = D7	FC MFM
FD	FD mit CLK = FF	FD MFM
FE	FE mit CLK = C7, CRC löschen	FE MFM
FF	FF mit CLK = FF	FF MFM

* 4,5 Taktbit auslassen
** 3,4 Taktbit auslassen

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	FF (or 00)[1]
6	00
1	FC (Index Mark)
26	FF (or 00)
6	00
1	FE (ID Address Mark)
1	Track Number
1	Side Number (00 or 01)
1	Sector Number (1 thru 1A)
1	00
1	F7 (2 CRC's written)
11	FF (or 00)
6	00
1	FB (Data Address Mark)
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
27	FF (or 00)
247**	FF (or 00)

*Write bracketed field 26 times

**Continue writing until FD179X interrupts out. Approx. 247 bytes.
1-Optional '00' on 1795/7 only.

Bild 7. IBM-Formatierung mit SD. Nach dem Befehl Schreibe Spur müssen dem Controller auf Anforderung diese Bytes der Reihe nach zur Verfügung gestellt werden

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
80	4E
12	00
3	F6
1	FC (Index Mark)
50*	4E
12	00
3	F5
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	Side Number (0 or 1)
1	Sector Number (1 thru 1A)
1	01
1	F7 (2 CRCs written)
22	4E
12	00
3	F5
1	FB (Data Address Mark)
256	DATA
1	F7 (2 CRCs written)
54	4E
598**	4E

*Write bracketed field 26 times
**Continue writing until FD179X interrupts out. Approx. 598 bytes.

Bild 8. Bei MFM gehen genau doppelt so viele Bytes in eine Spur

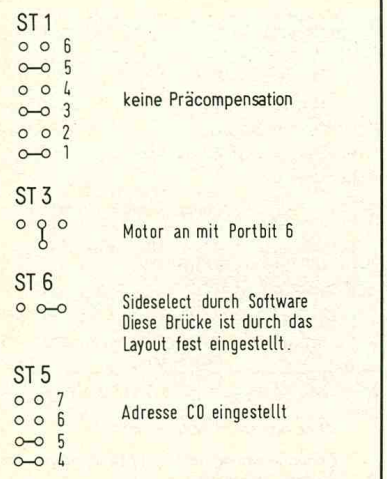


Bild 9. Die Belegung der Brücken auf FLO2

Nach dem Einschalten darf zunächst nichts weiter passieren, das Grundprogramm, bzw. der Monitor müssen sich melden.
Nun gebe man den Wert 55 auf IO-Port C1 aus. Entweder mit dem Befehl „IO-Setzen" oder mit „QOC1 55" beim mc-Monitor.
Bild 10. Wenn man nun mit „IO-Lesen" oder „QIC1" den Inhalt des Ports wieder abfragt, so muß 55 wieder erscheinen, wie *Bild 11* zeigt.
Nun führe man den Test noch mit dem Wert AA aus. Man hat damit geprüft, ob alle Datenleitungen o. k. sind und ob die Adressierung stimmt. Jetzt geht es an das Laufwerk. Das Laufwerk (Mini oder Maxi) wird über ein Flachbandkabel an die entsprechende Stiftleiste der FLO2-Baugruppe angesteckt. Dazu sollte man zuvor die Anleitung des Laufwerks studieren, um alle Einstellungen beim Laufwerk richtig setzen zu können. Das Laufwerk muß auf Laufwerksadresse 1, DS1, A, also als das erste eingestellt werden. Das Laufwerk benötigt eigene Versorgungsspannungen. Bei Minilaufwerken werden neben +5 V meist noch +12 V und bei Maxilaufwerken neben +5 V

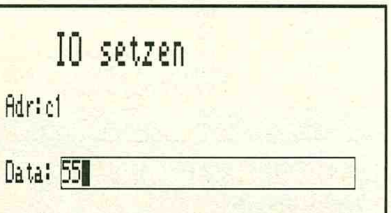


Bild 10. Ein Test mit dem Port C1, auf dem NRD-Klein-Computer gefahren

noch +24 V, und ggf. einige andere Spannungen verlangt.
Jetzt kann der Test beginnen. Zunächst wird eine formatierte Diskette eingelegt. An Port C4 wird der Wert 11h bei Maxilaufwerken mit einfacher Dichte ausgegeben, bei Minilaufwerken der Wert 21h für doppelte Dichte oder 31h für einfache Dichte (siehe Bild 3). Danach wird der Wert 0Fh auf Port 0C0h ausgegeben. Damit wird ein Restore-Befehl ausgeführt. Das Laufwerk muß nun ansprechen (LED am Gehäuse leuchtet auf, falls vorhanden) und dann wieder ausgehen.

Bild 11. Das muß danach gelesen werden können

Nun gibt es verschiedene Fälle nach Einlesen von Port 0C0H. *Bild 12* zeigt ein Beispiel. Die Fehlerbelegung entspricht Tabelle 6. Hier also ein Beispiel, das o. k. ist. Bit 6 gesetzt bedeutet „Schreibschutz", da die Diskette schreibgeschützt war. Bit 5 bedeutet „Kopf liegt noch auf", da das Laufwerk noch selektiert war. Bit 2 auf 1 bedeutet „Spur 0", was auch o. k. ist, denn der Befehl sollte ja die Spur 0 anfahren. Wenn man etwas später wieder abfragt, so erscheint der Wert 0 in allen Bits, denn das Laufwerk wird nur für eine bestimmte Zeit selektiert. Fehler werden signalisiert, wenn andere Bits gesetzt sind. Bit 7 würde zum Beispiel bedeuten „Laufwerk nicht Ready". Vielleicht steckt dann die Diskette falsch im Laufwerk, ihr „Label" muß zum Verschlußhebel zeigen (je nach Laufwerk). Bit 0 kann gesetzt bleiben, wenn z. B. der Indexpuls nicht kommt. Man kann das einmal probieren, indem man die Diskette aus dem Laufwerk herausnimmt und dann den Befehl 0Fh an Port 0C0h ausgibt. Wenn Bit 4 oder 3 gesetzt sind, so liegt ein Lesefehler vor. Entweder stimmt dann die eingestellte Dichte nicht, oder die Diskette ist nicht formatiert (z. B. bei manchen käuflichen Minidisketten) oder die Schaltung arbeitet nicht einwandfrei. Man

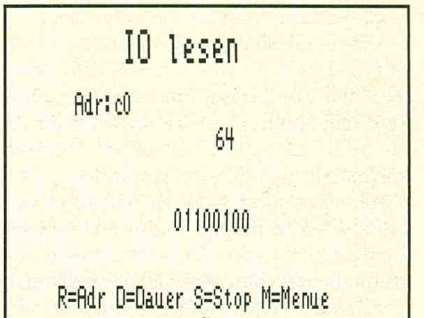


Bild 12. Ein mögliches Ergebnis beim Restore-Befehl

sollte dann neben einer Kontrolle der einzelnen Lötstellen auch die Diskette überprüfen. Aufschluß ergibt (für Experten) dann ein Oszillogramm der Signale RAWRD, RCLK am Controller-IC und $\overline{\text{RAWRD}}$ am Controller-IC (IC 4). Auch könnte der Takt fehlen (XTAL Pin 11 IC 3, oder CLK Pin 24 IC 4). Nun ein weiterer Test: Der Such-Befehl. Damit kann der Schrittmotor kontrolliert werden. Auf Port 0C4h gebe man wieder den Laufwerkscode (Maxi SD = 11h, Mini DD = 21h), auf Port 0C3h gebe man die Spurnummer, z. B. 20h und danach auf Port 0C0h den Wert 1Fh, was dem Seek-Befehl entspricht. Jetzt muß das Laufwerk sich angesprochen fühlen und der Kopf sollte in eine innere Spur wandern. Anschließend kann man wieder den Status an Port 0C0h kontrollieren. Dort muß nach einigen Augenblicken der Wert 0 stehen oder, wenn man schnell genug mit der Abfrage ist, der Wert 20h oder 60h.
Zum Test unter dem 68008/68000 noch ein Hinweis. Beim 68008 werden die Adressen $FFFFFFC0 bis $FFFFFFC4 für den Controller verwendet und beim 68000 die Adressen $FFFFFFC0*2 bis $FFFFFFC4*2, da beim 68000 alle Systemports auf geraden Adressen liegen müssen.
Nachdem mit diesem Test auch das Lesen geprüft wurde, ist der Gesamttest schon fast beendet. Noch nicht getestet ist die Interrupt-Logik, die aber kaum Schwierigkeiten macht.
Jetzt müßte man das Betriebssystem laden können und loslegen. Beim 68000/8 Grundprogramm wähle man dazu das Menü Floppy-Start (ab Version 4.0). Beim mc-CP/M-Computer wird mit dem I-Befehl gearbeitet. Beim NDR-Klein-Computer erfolgt der Start mit Hilfe des neuen Programms FLOMON, das dazu auf der BANK/BOOT-Karte untergebracht ist. Für CP/M benötigt man beim Z80 64 KByte RAM und beim 68000/8 mindestens 128 KByte.

Rolf-Dieter Klein

Flomon, das Z80-Monitorprogramm für die Floppy

Flomon sitzt in einem 8-KByte-EPROM auf der Bank/Boot-Baugruppe, wie im letzten Abschnitt beschrieben. Flomon beinhaltet Unterprogramme für die Floppy, wie Lesen und Schreiben, sowie Unterprogramme für die Bildschirmverwaltung. Dabei sind aber auch komfortable Grafik-Befehle eingebaut, die mit Steuerzeichen erreichbar sind. Dadurch ist es möglich, auch von beliebigen Programmiersprachen aus auf die leistungsfähige Grafik der GDP-Baugruppe zuzugreifen.

Wenn man den Z80-Computer mit Flomon startet, so meldet er sich wie in *Bild 1* gezeigt. Man hat dann vier Auswahlmöglichkeiten. Wenn die Taste 1 drückt, so wird die Floppy gestartet und davon das CP/M-Betriebssystem geladen. Welches Laufwerk angeschlossen ist, erkennt der Rechner automatisch. Damit kann man Laufwerke mit 8 Zoll, 5¼ Zoll, 3½ Zoll und 3 Zoll betreiben, alles mit einfacher oder doppelter Dichte. Allerdings muß dazu auf der Floppy das dazugehörige und angepaßte CP/M-Betriebssystem vorhanden sein. Dazu im nächsten Abschnitt mehr.
Mit Taste 2 kann man das Programm auf der Adresse E0000h, also Bank E, Adresse 0, starten. Damit kann man sich dort eine Sammlung von häufig gebrauchten Programmen (in EPROMs) anlegen. Mit der Taste 3 läßt sich der zweite EPROM-Platz auf der Bank-/Boot-Baugruppe starten, dort kann man z. B. das Grundprogramm (Version auf Adresse 2000h) starten. Mit den Tasten CTRL und C gelangt man in den Testmodus und kann die Befehle des Flomon prüfen, die nun besprochen werden sollen.

```
FLOMON 1.5 Rolf-Dieter Klein (C) 1984

1 = Floppy Boot
2 = Go E0000h
3 = Go Bank 2000h
CTRL-C=Testmode
```

Bild 1. So meldet sich Flomon

ASCII ein

Dazu drücken Sie bitte die Taste CTRL und dann die Taste „C". Es passiert zunächst nichts weiter, der Cursor blinkt noch. Wenn man nun aber einen Buchstaben drückt, zum Beispiel „A", so erscheint dieser auf dem Bildschirm. Sie befinden sich jetzt im sogenannten Alpha-Modus, indem die von der Tastatur kommenden Byte als ASCII-Zeichen interpretiert werden. Die Werte von 20h bis 7Fh sind druckende ASCII-Zeichen, die auf dem Bildschirm erscheinen. Die Codes 0 bis 1Fh sind nicht druckende Steuerzeichen. Davon dürften zumindest CR und LF schon bekannt sein. Sie haben den Wert 0Dh und 0Ah. Wenn man CR drückt, so wird der Cursor an die linke Seite des Bildschirms gerückt. Bei LF wandert eine Zeile nach unten. Es gibt beim Flomon viele solcher Zeichen. *Bild 2* zeigt die vollständige Auflistung aller vorhandener Befehle. Die Zeichen 8, 9, 0Ah, 0Bh, 0Ch, 0Dh, 16h, 1Ah, 1Eh werden direkt interpretiert.

```
BEFEHLSBESCHREIBUNG  (C) 1984 Rolf-Dieter Klein
-----------------------------------------------

 NDR-Klein-Computer FLOMON 1.5

Befehle
=======

ALPHA-MODUS
-----------

Ist nach dem Einschalten aktiv
20h..7fh                    Codes der sichtbaren ASCII-Zeichen

08h      CTRL H             Backspace
09h      CTRL I             Cursor Right
0ah      CTRL J             Linefeed
0bh      CTRL K             Cursor Up
0ch      CTRL L             Cursor Right
0dh      CTRL M             Carriage Return
16h      CTRL V             Cursor Down
1ah      CTRL Z             Clear
1eh      CTRL ^             Home

Escape Sequenzen

1bh 3dh y x   ESC = r c     Cursor setzen (y,x =20h..7fh)

1bh 44h n     ESC D n       Lokal Mode einschalten.
                            ESC D L, schaltet den Mode ein
                            ESC D N, schaltet wieder aus.

1bh 51h       ESC Q         Zeichen bei Cursorposition einfügen

1bh 57h       ESC W         Zeichen bei Cursorposition löschen

1bh 45h       ESC E         Zeile bei Cursorposition einfügen

1bh 52h       ESC R         Zeile bei Cursorposition löschen
```

Bild 2. Die Befehle von Flomon

1bh 54h	ESC T	Zeile ab Cursorposition bis zum Ende löschen
1bh 74h	ESC t	Zeile ab Cursorposition bis zum Ende löschen
1bh 59h	ESC Y	Seite ab Cursorposition bis Ende löschen
1bh 79h	ESC y	Seite ab Cursorposition bis Ende löschen
1bh 7ah x	ESC z n	n='0' amerikanischer Zeichensatz; n='1' deutscher Zeichensatz

Doppelescape Sequenzen

1bh 1bh 47h	ESC ESC G	Grafik-Modus

GRAFIK-MODUS

Das Zeichen n steht für die Eingabe eines numerischen Wertes. n kann eine vorzeichenbehaftete Dezimalzahl sein oder eine sedezimale Zahl: 0, -50, $FF, -$4034. Mehrere numerischen Werte werden durch Leerzeichen getrennt. Ferner sind %dx %dy relative Koordinaten mit Prozentzeichen bei den Befehlen D,M,L,l moeglich.

CR ist das Abschlußzeichen. Es kann bis auf wenige Ausnahmen auch das Zeichen Strichpunkt (';') verwendet werden.

A	in den Alpha-Modus zurückschalten,
M n1 n2 CR	Positionieren auf x=n1 y=n2
D n1 n2 CR	Vektor nach x=n1, y=n2 zeichnen
m b1 b2 b3 b4	Wie M, jedoch mit binärer Übertragung und daher schnell. x = b1,b2 (b1 ist höherwertiges Byte, b2 ist niederwertiges Byte) y = b3,b4
d b1 b2 b3 b4	Wie D, jedoch binär. Format wie bei m
J n1 n2 CR	Wie D, jedoch mit Angabe von relativen Koordinaten dx = n1 dy = n2
P n CR	Seite asynchron anwählen. n = Nummer der Schreibseite * 4 + Nummer der zu lesenden Seite z.b. Schreibseite =3 , Leseseite = 1 n = 3*4 + 1 = 13. Befehl: P 13 CR
S n CR	Seite synchron anwählen. n wird wird wie oben bestimmt, die Seite wird jedoch synchron zum Bildwechsel umgeschaltet.
X n CR	Die Seiten 0,1,2,3 werden zyklisch angezeigt, n gibt die Sichtdauer einer Seite an. Die Anzeigedauer beträgt n * 20ms Wenn man den Befehl gegeben hat, so wird auch der Alpha-Mode umgeschaltet, wenn man mit A zurückkehrt, bis entweder ein Bildschirmlöschen im Alpha-Mode stattfindet oder X und Y im Graphikmode ohne Parameter aufgerufen wurden, also z.B. X cr. n = 0 beendet den Wechsel
Y n CR	Es werden jeweils nur zwei Seiten zyklisch angezeigt und zwar die Seiten 0 und 1, wenn eine dieser beiden Seiten als Leseseite definiert war, sonst 2 und 3, für den Alphamode gilt das Gleiche, wie bei X. Wenn man z.B. X0 und CR eingibt, so wird das automatische Seiten-umschalten ausgeschaltet, auch im Alpha-Mode und man Graphik und Text einfach mischen. Der Cursor bleibt dann allerdings als heller Block und blinkt nicht mehr.
C	Löschen der aktuellen Schreibseite
Z	Löschen aller Seiten 0 bis 3. Die zuletzt angewähle Bildschirmseite wird danach eingestellt.
G n1 n2 CR	Befehl an GDP n1 = Nummer des GDP-Ports (0..15), n2 = Datenwert an diesen Port
B text CR	Text (20h..7fh) an GDP-Port 0 senden. Der Text wird wird ab aktueller Koordinate des GDP ausgegeben
V binär 00h	Binärdaten an den GDP-Port 0 senden. Das ASCII-Zeichen NUL beendet die Übertragung
O n1 n2 n3 n4 CR	Ellipsenabschnitte zeichnen. Mit n1 wird die Länge der Halbachse in x-Richtung angegeben, mit n2 die Länge der Halbachse in y-Richtung. Mit n3 wird der Startwinkel bezüglich der x-Achse in Grad angegeben. Mit n4 der Endwinkel des Ellipsenabschnittes. Der Ellipsenabschnitt wird von der aktuellen x,y-Koordinate bis zum Erreichen des Endwinkels gezeichnet. Der Ellipsenmittelpunkt wird aus der Startwinkel- und Halbachsen-angabe vor Beginn des Zeichnens automatisch errechnet.

```
O n1 n2 n3 n4 1 CR
```
Wie oben oben, jedoch der vom Kurvenstück und den Radien zum Mittelpunkt begrenzte Raum gefüllt (Torte)

```
R n1 n2 CR
```
Rechteck ab aktueller x,y-Koordinate. n1=dx und n2=dy geben die Breite und die Höhe des Rechtecks an

```
R dx dy 1 CR
```
Rechteck gefuellt zeichnen. Sonst wie oben.

```
L n1 n2 n3 n4 ... nn nm CR
```
Polygon zeichnen, mit absoluten Koordinaten.
x0=n1 y0=n2 gibt die Startposition an, alle weiteren Paare geben die Eckpunkte des Polygons an.
Der letzte Eckpunkt wird wieder mit dem Startpunkt verbunden

```
1 n1 n2 n3 n4 n5 n6 CR
```
Dreieck gefüllt zeichnen
x0 = n1 y0 = n2
x1 = n3 y1 = n4
x2 = n5 y2 = n6

```
F n1 n2 n3 CR
```
Fadenkreuz zeichnen, an Position x=n1 y=n2, auf Seite n3 (0..3).
Altes Fadenkreuz wird gelöscht. Die Schreib- und Leseseite bleiben erhalten.

```
F n1 n2 n3 rmw  CR
```
Fadenkreuz setzen, mit RMW
rmw=0 kein RMW-Zyklus
rmw=1 mit XOR-Mode
Nur mit Zusatzhardware auf der GDP verwendbar.

```
r rmw CR
```
rmw=0 normal Mode
rmw=1 mit XOR-Mode arbeiten, alle Befehle sind betroffen. Achtung, nur mit Zusatzhardware. Bit 0 am Port 60h bestimmt den RWM-Mode.

```
r CR
```
rmw=0 setzen

```
T
```
Koordinate auf Stack. Die aktuellen Koordinaten des GDP X und Y werden auf einen Stack gelegt. Bei zuvielen Koordinaten kommt es zum Überlauf. Die Größe ist vom Restspeicher abhängig.

```
U
```
Koordinate von Stack. Die zuletzt mit T abgespeicherten Koordinaten werden zurückgeladen und die GDP-Position mit MOVETO eingestellt. Sind keine Koordinaten im Stack, so bleibt die Position unverändert.

```
Q
```
Stack loeschen. Der Koordinatenstack wird gelöscht. Dies ist zu Programmbeginn nützlich, um zu vermeiden, daß sich Koordinaten auf dem Stack ansammeln.

```
WA string CR
```
Symbol für den Fadenkreuz-Befehl umdefinieren.
string ist eine Zeichenkette mit Zeichen im ASCII-Bereich (30h,31h,40h..5fh).
0 (30h) = Schreibstift hoch,
1 (31h) = Schreibstift runter.
Der Code für die Schreibstiftbewegung berechnet sich wie folgt:
Richtung + (8 * Länge) + 40h
Es stehen die Richtungen von 0 bis 7 (* 45 Grad) zur Verfügung

```
WA CR
```
Symbol für den Fadenkreuzbefehl auf das Symbol "Fadenkreuz" zurücksetzen

```
WB
```
Fadenkreuzsymbol an aktueller x,y-Koordinate auf der aktuellen Schreibseite setzen

```
WC n1 n2 CR
```
Fadenkreuzsymbol vergrößern und drehen.
n1 = Vergrößerungsfaktor (1..255)
n2 = Drehung (0..7)

```
WC CR
```
Rückstellen auf n1 = 1, n2 = 0

```
WD n1 n2 n3 ... nn CR
```
Download. Es können die Daten n2 bis nn ab Adresse n1 in den Arbeitsspeicher geladen werden. Dabei kann man auf das normalerweise verborgene RAM auf der BANK/BOOT-Karte zugreifen.

```
WE n1 CR
```
Programm auf Adresse = n1 starten

```
H y0 CR
```
Füllen entlang der horizontalen Achse. Alle Vektorbefehle sind betroffen. Von da an wird die Fläche unter Vektoren bis zur angegebenen y-Achse ausgefüllt. (Befehl D,J und O sind betroffen).

```
H CR
```
Füllen ausschalten

```
I x0 CR
```
Füllen entalng der vertikalen Achse, wie H, jedoch mit X-Achse.

```
I CR
```
Füllen ausschalten

Es folgen noch einige Befehle für die Farberweiterung. Dabei wird über einen zusätzlichen Farbport (adresse 0A0h) der Farbcode ausgegeben. Die Befehle wurden schon eingebaut, um möglichst frühzeitig Kompatibilität zu erreichen.

```
WG farbe CR      Farbcode setzen
                 farbe: 0..255

7       6        5        4        3        2        1        0

DI weiß DW weiß DI blau DW blau DI grün DW grün DI rot  DW rot

0= aktiv geschaltet.
mit DW wird der Schreibvorgang eingeschaltet (=0)
mit DI kann Löschen je Farb-Ebene eingestellt werden (=1)

wichtige Codes fuer dominantes Schreiben:
Hintergrund:     Code    3F
weiß:            Code    C0
std:             Code    00
blaugruen:       Code    C3
violett:         Code    CC
blau:            Code    CF
gelb:            Code    F0
gruen:           Code    F3
rot:             Code    FC

K seite-A Farbe-A pen-A seite-B Farbe-B pen-B

                 Mehrseitenschreiben
                 Bereich seite: 0..3
                 Farbe = Farbcode
                 pen =0 bei schreiben =1 bei loeschen

K CR             stellt Normalzustand her

alle Linienzeichenbefehle und Zeichenausgabe sind betroffen


Monitoreinsprünge bei FLOMON -- Graphik-Routinen

F040:   JP CLRALL        ; alle Seiten loeschen
F043:   JP CLRINVIS      ; aktuelle WRT-Seite loeschen
F046:   JP MOVETO        ; hl=x de=y
F049:   JP DRAWTO        ; hl=x de=y
F04C:   JP WRTPG         ; c=Schreibseite
F04F:   JP VIEWPG        ; c=Leseseite
F052:   JP RMWPG         ; c=RMW-Mode (0,1)
F055:   JP WAIT          ; warten bis GDP fertig
F058:   JP CMD           ; c=Befehl fuer GDP
```

```
    Line 1   Col 1   Insert   Indent  A:TEST.PAS
|

program flomon(output);
begin
 writeln(chr(27),chr(27),'G');
 writeln('ZP0');
 writeln('M300 150');
 writeln('R100 50 1');
 writeln('O-100 50 0 270 1');
 writeln('F200 20 0');
 writeln('10 0 100 0 0 100');
 writeln('M20 200');
 writeln('G3 $44');
 writeln('BHallo');
 writeln('V0');
 writeln('A');
end.
```

Bild 4. Ein Testprogramm im Turbo-Pascal

Steuern mit ESC

Mit „ESC = ! !“ zum Beispiel kann man den Cursor in die erste Zeile und dort in die erste Spalte positionieren.
Die Steuersequenzen wurden nicht willkürlich gewählt, sondern entstammen einer in den USA verbreiteten Terminalgeneration der Firma Televideo: TVI 912, 920 und 950. Man kann so CP/M-Standard-Programme wie Wordstar verwenden, die diese Terminals meist direkt unterstützen.
Neben dem reinen Alpha-Modus gibt es aber auch noch einen speziellen Graphik-Modus. Dorthin gelangt man wenn man zweimal ESC drückt und dann Shift und G, also großes G tippt. Mit Shift A, oder ESC ESC Shift A kommt man wieder in den alphanumerischen Betrieb zurück. Es gibt eine Vielzahl von Grafik-Befehlen. Wenn man in den Gra-

Dann gibt es sogenannte Escape-Sequenzen. Dabei wird zunächst das Zeichen Escape (Taste mit der Beschriftung ESC oder CTRL und „eckige Klammer auf“) gedrückt und dann ein weiteres Zeichen ESC und R (Achtung großes R, immer wie im Bild 2 angegeben) löscht zum Beispiel die Zeile, in der der Cursor steht. Ich empfehle jedem zunächst einmal alle Kombinationen einfach auszuprobieren, um ein Gefühl für den Befehlssatz zu gewinnen.

Bild 3. Eine Grafik-Demonstration

```
     Line 20   Col 6   Insert   Indent  A:TEST.PAS
program flomon(output);
var x,y:integer; ch:char;
 key : text;
begin
 assign(key,'KBD:'); reset(key); { fuer Einzelzeichen ohen Bildschirmausgabe}
 writeln(chr(27),chr(27),'G');   { Graphik-Mode einschalten }
 writeln('ZP0;V1;M0 0;R20 10');  { Wechselfrequenz und kleines Rechteck }
 x:=256; y:=128;                 { Startposition des Fadenkreuzes }
 repeat
  writeln('F',x,' ',y,' 1');     { Fadenkreuz ausgeben }
  writeln('G0 128');             { dort immer einen Punkt setzen }
  read(key,ch); case ch of       { ein zeichen einlesen, ohne Bildschirmausgabe
   '+': y:=y+5; '-': y:=y-5;     { +- und <> steuern das Fadenkreuz }
   '<': x:=x-10; '>': x:=x+10;
  end;
 until (x<20) and (y<10);        { Stop , wenn im kleinen Rechteck angekommen }
 writeln('A');                   { zurueck in Alpha-Mode, nicht vergessen ! }
end.
```

Bild 5. Der Fadenkreuz-Befehl in Turbo-Pascal

fik-Modus kommt, verschwindet zunächst der Cursor. Der alte Text aus dem Alpha-Modus bleibt jedoch erhalten. Nun kann man neue Befehle geben, z. B. folgende Sequenz:

```
M0 0          carriage return
D300 150      carriage return
```

carriage return ist die Taste CR. Wenn man diese Zeichen eingibt, ändert sich auf dem Bildschirm zunächst nichts. Erst nach der letzten Eingabe erscheint eine schräge Linie auf dem Bildschirm. Jetzt versuche man einmal folgendes:

```
ZP0
M300 150                carriage return
R100 50 1               carriage return
O-100 50 0 270 1        carriage return
F200 20 0               carriage return
l10 0 100 0 0 100       carriage return
M20 200                 carriage return
G3 $44                  carriage return
BHallo                  carriage return
```

Bild 3 zeigt das Ergebnis, wenn Sie alles richtig getippt haben. Probieren Sie auch die anderen Befehle aus Bild 2. All diese Befehle kann man, wie schon versprochen, von verschiedenen Sprachen aus verwenden, denn sie sind ja nur einfache Ausgaben an den Grafikprozessor. So zeigt *Bild 4* ein Beispiel in Turbo-Pascal. Der Befehl Y0 bewirkt, daß das Grafik-Bild nicht blinkt, denn in Alpha-Modus werden normalerweise zwei Bildseiten verwendet, die im Cursortakt umgeschaltet werden. Mit Y0 kann man das abstellen. Dann blinkt allerdings auch der Cursor nicht mehr. Wenn man nur Y und CR eingibt erhält man wieder den üblichen Umschaltwert. Mit Y1 ergibt sich eine ganz hohe Umschaltfrequenz. Die Befehle Y und X kann man für einen automatischen Bildwechsel verwenden (was z. B. für Phasenzeichnungen interessant ist).

Bild 6.
Ein Ausgabebeispiel

Es gibt übrigens auch ein Fadenkreuz (der Befehl F), das normalerweise auf Seite 1 ausgeben wird und dann mit Y1 dem normalen Bild überlagert werden kann. Damit kann man einfache interaktive Eingabeprogramme schreiben.
Bild 5 zeigt ein Programmbeispiel. Und *Bild 6* eine mögliche Ausgabe davon.
Damit das CP/M, genauer gesagt, das BIOS auf Flomon zugreifen kann, gibt es ein paar Einsprungstellen, die *Bild 7* zeigt. Diese Einsprünge beinhalten Consol-Ein- und Ausgabe, so wie auch die Floppy-Routinen. Ferner gibt es auch für die Grafik noch direkte Einsprünge, wenn man besonders schnellen Zugriff auf die Grafik braucht (doch das nur für Spezialisten).
Der Einsprung FLOPPY ist wohl der wichtigste; wie er funktioniert, erfahren Sie im Abschnitt über das BIOS und das CP/M.

Flomon–Einsprünge

```
F000    JP START        Start des Flomon–Programms

F003    JP CI           Ein Zeichen von der Tastatur lesen.
                        Das Zeichen kommt in Register A an.
F006    JP RI           Ein Zeichen von der CAS–Schnittstelle lesen.
                        Das Zeichen kommt in A an.
F009    JP CO           Ein Zeichen auf dem Bildschirm ausgeben.
                        Das Zeichen steht dazu im Register C.
F00C    JP PO           Ein Zeichen auf die CAS–Schnittstelle ausgeben
                        Das Zeichen steht dazu im Register C.
F00F    JP LO           Ein Zeichen auf die Druckerschnittstelle ausgeben.
                        Das Zeichen steht im Register C.
F012    JP CSTS         Consolstatus prüfen. Im Register A steht der
                        Wert 0FFh, wenn ein Zeichen ansteht, sonst der
                        Wert 0. Das Zeichen wird erst durch CI tatsächlich
                        gelesen.
F015    JP IOBYTE       Nicht implementiert. Jedoch aus komp. Gründen
                        belassen, liefert den Wert 0 in A.
F018    JP IOSET        Nicht implementiert.
F01B    JP MEMCHK       A=EFh und B=FF, letzte verfügbare RAM–Zelle.
F01E    JP START        Neustart des Flomon.
F021    JP FLOPPY       Universal Floppy–Einsprung Mini/Maxi.
F024    JP MAXI         mit MAXI–Voreinstellung, mc–kompatibel.
F027    JP MINI         mit MINI–Voreinstellung, mc–kompatibel.
F02A    JP WINCHESTER   für spätere Erweiterung mit Winchester Laufwerk.
;
        DEFW 0          0=keine Tabelle vorhanden.
        DEFW LASTMON    letzte Speicherzelle im Monitorprogramm
        DEFW FREEMEM    dort sicher Platz fuer eigene Buffer.
        DEFS 13         Reserve fuer Erweiterungen
; Grafik–Befehle für schnelle Grafiken
;
F040    JP CLRALL       alle Bildseiten löschen
F043    JP CLRINVIS     unsichtbare Bildseite löschen
F046    JP MOVETO       HL=X,DE=Y, Positionieren
F049    JP DRAWTO       HL=X,DE=Y, Linie zeichnen
F04C    JP WRTPAGE      C=Schreibseite 0,1,2,3
F04F    JP VIEWPAGE     C=Leseseite 0,1,2,3
F052    JP RMWPAGE      C=0, normal, =1 mit XOR–Mode, nur mit Zusatzhardw.
F055    JP WAIT         Wartet bis der GDP fertig ist
F058    JP CMD          C=Befehl, wird an den GDP ausgegeben
;
; Spezialbefehle
;
F05B    JP BANK         HL=Adresse in der Quellbank 0..EFFF
                        DE=Adresse in der Zielbank  0..EFFF
                        C=Nummer der Quellbank      0..F
                        B=Nummer der Zielbank       0..F
                        Es werden 128 Bytes transportiert.
                        Dabei muß von Adresse F000 bis FFFF
                        auf der jeweiligen Bank auf jeden Fall
                        ein RAM–Speicher sein. Sonst wird
                        ein Carry als Ergebnis geliefert.
                        Der Bereich F000 bis FFFF ist immer
                        für das Transportprogramm reserviert.
```

Bild 7.
Die Flomon-Einsprünge

Rolf-Dieter Klein

Das BIOS für FLO-2

CP/M für den NDR-Klein-Computer

Im letzten Kapitel haben Sie alles über die Hardware der neuen Floppy-Karte für den NDR-Computer und den mc-CP/M-Computer nachlesen können. Jetzt folgt die Beschreibung des CP/M-2.2 für den Vollausbau des Rechners mit der Z80-CPU. Vieles, was hier gesagt wird, ist auch für den 68008-Ausbau schon gültig.

CP/M ist die Abkürzung für Control Program for Microcomputers, zu deutsch Steuerprogramm für Mikrocomputer. Das sagt noch nicht viel, denn das Grundprogramm des NDR-Klein-Computers steuert auch eine ganze Menge. Was CP/M so wertvoll macht, ist die Bedienung eines Floppy-Laufwerks. CP/M hat als eine seiner Hauptaufgaben die Anordnung der Daten auf einer Diskette zu verwalten. CP/M ist also ein sogenanntes Dateiverwaltungssystem. Denn wenn man so viel Speicherplatz wie auf einer Diskette besitzt, muß man schnell und sicher auf die Daten zugreifen können. Dazu muß der Computer immer aktuell wissen, wo welche Daten und Programme mit welchem Namen auf der Diskette stehen. Das wird in dem Inhaltsverzeichnis einer jeden Diskette, dem Directory (oft Katalog genannt) mitnotiert. Dort sind die Namen aller Dateien verzeichnet, wie auch deren Blocknummern, woraus CP/M Spur und Sektor, also den physikalischen Ort der Datei und ihre Größe berechnen kann. Wenn zum Beispiel Daten oder Programme gelöscht werden sollen, so muß CP/M dafür sorgen, daß die Blöcke, in welchen der zu löschende Eintrag steht, wieder für neue Programme verfügbar werden. Man muß also Dateien anlegen können, man muß sie löschen können, oder ein Inhaltsverzeichnis ausgeben können. All dies sind Aufgaben eines Betriebssystems, wie CP/M eines ist.

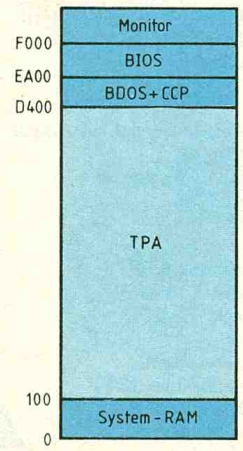


Bild 1. Die Speicheraufteilung unter CP/M

CP/M ist anpaßbar

CP/M hat aber auch noch eine weitere interessante Eigenschaft. Man kann es relativ einfach an die verschiedensten Gegebenheiten anpassen. Diese Eigenschaft war es, die CP/M so verbreitet hat. Weltweit gibt es wohl 10 Millionen Benutzer von CP/M-Rechnern. Deshalb gibt es für CP/M nahezu beliebig viel Software, von Programmiersprachen wie Pascal, Fortran, APL, C, Lisp, Algol, Forth oder Basic angefangen, über zahlreiche Programme aus der Textverarbeitung (Wordstar oder Kalkulationsprogramme wie Calcstar) bis zu Datenbankprogrammen oder höchstwertigen Spezial-Utilitys. Diese Anpaßfähigkeit war auch für uns ausschlaggebend. Denn CP/M läßt sich auch leicht an den NDR-Klein-Computer anpassen.

Zunächst noch etwas Allgemeines. *Bild 1* zeigt die Speicheraufteilung unter CP/M. Von Adresse 0 bis 100h liegt der System-Speicher, den CP/M teilweise für eigene Zwecke nutzt. Ab 100h beginnt die sogenannte TPA (Transient Program Area). Sie nimmt die Anwenderprogramme auf. Ab Adresse D400h beginnt das eigentliche CP/M, wenn es so wie unseres angepaßt wird. Zunächst folgt ein Systemprogramm namens CCP (Consol Command Prozessor). Es hat die Aufgabe, Befehle von der Tastatur entgegenzunehmen und auszuwerten. Dann folgt das BDOS, was die eigentliche Diskettenverwaltung durchführt. Ab Adresse EA00 folgt das BIOS, das schließlich den anpaßbaren Teil, das System-Interface darstellt. Das BIOS muß man selbst schreiben, wenn man kein fertiges CP/M bezieht. Dieses BIOS ist in *Bild 2* gezeigt, so wie man es für den NDR-Computer mit Z80-CPU gebrauchen kann. Das BIOS selbst greift wieder auf weitere Routinen zu, die bei uns ab Adresse F000h stehen und im EPROM mit FLOMON untergebracht sind. Dort stehen dann die eigentlichen Floppy-Unterprogramme, die den Zugriff auf die Diskette steuern. Diese Routinen erwarten Spur- und Sektor-Nummern, sie wollen nichts mehr von Dateinamen wissen. Wie gesagt, die Umsetzung der Namensangabe einer Datei, verbunden mit dem Schreib-, Lese- oder Lösch-Wunsch, in die Nummern der zu bearbeitenden Floppyscheibenbereiche, das ist die Aufgabe des BDOS. In das BIOS ist eingearbeitet, welche Laufwerke am Rechner angeschlossen sind. Da man an die FLO2 alle möglichen Laufwerke anschließen kann, würde das BIOS sehr umfangreich werden, wenn man alle Laufwerkparameter unterbringen wollte.

Das neue mc-BIOS

Mit dem neuen BIOS kann man zunächst 5¼-Zoll-Laufwerke mit 80 Spuren, zwei Seiten und doppelter Aufzeichnungsdichte ansteuern. Man erhält dabei eine Speicherkapazität von etwa 780 KByte (abzüglich Inhaltsverzeichnis). CP/M selbst ist auf den äußeren Spuren der Diskette untergebracht. Es sind dafür 20 KByte reserviert, wobei das CP/M-2.2 allein nicht soviel benötigen würde. Da aber CP/M-68K für den 68008 und 68000 so viel Platz benötigt und Sie Ihren Computer ja einmal auf 16 Bit ausbauen wollen, seien etwa 10 KByte Systemplatz verschenkt. Sie können dann die Z80-Disketten auch mit dem 68000 vollständig lesen und geeignet bearbeiten. Insgesamt hat man mit unserem neuen Format wirklich ein Optimum an Kapazität erreicht.

```
                        MACLIB DISKDEF  ;LOAD DEFINTION FOR DISKS

                ;************************************************
                ; VERSION 3.1   VERSION 80 SPUR-LAUFWERK        *
                ; GEBOOTET WIRD VOM 80 SPUR-LAUFWERK            *
                ; A,B SIND DIE BEIDEN 80 SPUR LAUFWERKE         *
                ; C,D IST EIN 8ZOLL LW, C=VORDERSEITE,D=RUECKS. *
                ; E IST RAM-FLOPPY                             *
                ; (C) 1984 ROLF-DIETER KLEIN     841220       *
                ; ************************************************


0016 =          VERS    EQU     22      ; DEFINITIONEN, ALLEGEMEIN
FFFF =          TRUE    EQU     0FFFFH
0000 =          FALSE   EQU     NOT TRUE
FFFF =          TEST    EQU     TRUE


003C =          MSIZE   EQU 60          ; SPEICHERGROESSE, HIER 60K

A000 =          BIAS    EQU (MSIZE-20)*1024     ; MIN=20K
D400 =          CCP     EQU 3400H+BIAS  ; START DES CCP
DC06 =          BDOS    EQU CCP+806H    ; DORT BEGINNT DAS BDOS
EA00 =          BIOS    EQU CCP+1600H   ; UND DORT DAS BIOS
                ;
D400 =          CPMB    EQU CCP         ; START CP/M-BOOT.
                ;
1600 =          CPML    EQU BIOS-CPMB   ; LAENGE DES CP/MS
002C =          NSECTS  EQU CPML/128    ; ANZAHL DER BELEGTEN SEKTOREN
                ;
                ;
EA00                    ORG BIOS        ; START DES BIOS
                ;

0004 =          CDISK   EQU 4           ; ADRESSE IM SPEICHER, LETZTES LAUFWERK
0080 =          BUFF    EQU 80H         ; BUFFERADRESSE, DIE VOREINGESTELLT WIRD.
0005 =          RETRY   EQU 5           ; FEHLERVERSUCHE BEI BOOT ETC.

                ; VEKTORTABELLE DER BIOS-EINSPRUENGE

EA00 C3F2EA             JMP BOOT        ; KALT-START.
EA03 C314EB     WBOOTE: JMP WBOOT       ; WARM-START, BEI CTRL-C
EA06 C3E6EA             JMP CONST       ; CONSOL STATUS, ERGEBNIS IN A
EA09 C3E9EA             JMP CONIN       ; CONSOL EINGABE, NACH A
EA0C C3EFEA             JMP CONOUT      ; CONSOL AUSGABE VON C
EA0F C30FF0             JMP LIST        ; AUSGABE AUF DEN DRUCKER, VON C
EA12 C30CF0             JMP PUNCH       ; AUSGABE AUF PO, C-REGISTER
EA15 C306F0             JMP READER      ; EINGABE NACH RI, A-REGISTER

EA18 C3B2EB             JMP HOME        ; LAUFWERK NUR TRACK 0
EA1B C3B7EB             JMP SELDSK      ; LAUFWERK AUSWAEHLEN
EA1E C3CDEB             JMP SETTRK      ; SPUR AUSWAEHLEN
EA21 C3D2EB             JMP SETSEC      ; SEKTOR AUSWAEHLEN
EA24 C3EEEB             JMP SETDMA      ; ADRESSE FESTLEGEN
EA27 C301EC             JMP READ        ; SEKTOR LESEN
EA2A C377EC             JMP WRITE       ; SEKTOR SCHREIBEN
EA2D C3AFEB             JMP LISTST      ; DRUCKER FERTIG ?
EA30 C3D7EB             JMP SECTRAN     ; SEKTORUEBERSETZUNG
                ;

0004 =          OFFSET  EQU 4           ; FUER MINILW.

                ;
0185 =          DISKKAP EQU     389     ;TRUNC (5 * 1024 * (160-4)/2048) - 1
                ;                       ;CA. 780 K
                        DISKS 5         ; 5 LAUFWERKE HIERBEI
EA33+=          DPBASE  EQU     $       ;BASE OF DISK PARAMETER BLOCKS
EA33+00000000   DPE0:   DW      XLT0,0000H      ;TRANSLATE TABLE
EA37+00000000           DW      0000H,0000H     ;SCRATCH AREA
EA3B+05EE83EA           DW      DIRBUF,DPB0     ;DIR BUFF,PARM BLOCK
EA3F+B6EE85EE           DW      CSV0,ALV0       ;CHECK; ALLOC VECTORS
EA43+00000000   DPE1:   DW      XLT1,0000H      ;TRANSLATE TABLE
EA47+00000000           DW      0000H,0000H     ;SCRATCH AREA
EA4B+05EE83EA           DW      DIRBUF,DPB1     ;DIR BUFF,PARM BLOCK
EA4F+27EFF6EE           DW      CSV1,ALV1       ;CHECK, ALLOC VECTORS
EA53+A1EA0000   DPE2:   DW      XLT2,0000H      ;TRANSLATE TABLE
EA57+00000000           DW      0000H,0000H     ;SCRATCH AREA
EA5B+05EE92EA           DW      DIRBUF,DPB2     ;DIR BUFF,PARM BLOCK
EA5F+86EF67EF           DW      CSV2,ALV2       ;CHECK, ALLOC VECTORS
EA63+A1EA0000   DPE3:   DW      XLT3,0000H      ;TRANSLATE TABLE
EA67+00000000           DW      0000H,0000H     ;SCRATCH AREA
EA6B+05EE92EA           DW      DIRBUF,DPB3     ;DIR BUFF,PARM BLOCK
EA6F+B5EF96EF           DW      CSV3,ALV3       ;CHECK, ALLOC VECTORS
EA73+00000000   DPE4:   DW      XLT4,0000H      ;TRANSLATE TABLE
EA77+00000000           DW      0000H,0000H     ;SCRATCH AREA
EA7B+05EEBBEA           DW      DIRBUF,DPB4     ;DIR BUFF,PARM BLOCK
EA7F+DCEFC5EF           DW      CSV4,ALV4       ;CHECK, ALLOC VECTORS
                        DISKDEF 0,0,39,0,2048,DISKKAP,256,256,OFFSET ; OFFSET=4
EA83+=          DPB0    EQU     $               ;DISK PARM BLOCK
EA83+2800               DW      40              ;SEC PER TRACK
EA85+04                 DB      4               ;BLOCK SHIFT
EA86+0F                 DB      15              ;BLOCK MASK
EA87+00                 DB      0               ;EXTNT MASK
EA88+8401               DW      388             ;DISK SIZE-1
EA8A+FF00               DW      255             ;DIRECTORY MAX
EA8C+F0                 DB      240             ;ALLOC0
EA8D+00                 DB      0               ;ALLOC1
EA8E+4000               DW      64              ;CHECK SIZE
EA90+0400               DW      4               ;OFFSET
0000+=          XLT0    EQU     0               ;NO XLATE TABLE
                        DISKDEF 1,0
EA83+=          DPB1    EQU     DPB0    ;EQUIVALENT PARAMETERS
0031+=          ALS1    EQU     ALS0    ;SAME ALLOCATION VECTOR SIZE
0040+=          CSS1    EQU     CSS0    ;SAME CHECKSUM VECTOR SIZE
0000+=          XLT1    EQU     XLT0    ;SAME TRANSLATE TABLE
                        DISKDEF 2,1,26,6,1024,243,64,64,2 ; 8 ZOLL DEFINITION
EA92+=          DPB2    EQU     $               ;DISK PARM BLOCK
EA92+1A00               DW      26              ;SEC PER TRACK
EA94+03                 DB      3               ;BLOCK SHIFT
EA95+07                 DB      7               ;BLOCK MASK
EA96+00                 DB      0               ;EXTNT MASK
EA97+F200               DW      242             ;DISK SIZE-1
EA99+3F00               DW      63              ;DIRECTORY MAX
EA9B+C0                 DB      192             ;ALLOC0
EA9C+00                 DB      0               ;ALLOC1
EA9D+1000               DW      16              ;CHECK SIZE
EA9F+0200               DW      2               ;OFFSET
EAA1+=          XLT2    EQU     $               ;TRANSLATE TABLE
EAA1+01                 DB      1
```

Bild 2. Das ist das BIOS für den NDR-Computer mit Z80

```
EAA2+07                 DB      7
EAA3+0D                 DB      13
EAA4+13                 DB      19
EAA5+19                 DB      25
EAA6+05                 DB      5
EAA7+0B                 DB      11
EAA8+11                 DB      17
EAA9+17                 DB      23
EAAA+03                 DB      3
EAAB+09                 DB      9
EAAC+0F                 DB      15
EAAD+15                 DB      21
EAAE+02                 DB      2
EAAF+08                 DB      8
EAB0+0E                 DB      14
EAB1+14                 DB      20
EAB2+1A                 DB      26
EAB3+06                 DB      6
EAB4+0C                 DB      12
EAB5+12                 DB      18
EAB6+18                 DB      24
EAB7+04                 DB      4
EAB8+0A                 DB      10
EAB9+10                 DB      16
EABA+16                 DB      22
                        DISKDEF 3,2
EA92+=          DPB3    EQU     DPB2    ;EQUIVALENT PARAMETERS
001F+=          ALS3    EQU     ALS2    ;SAME ALLOCATION VECTOR SIZE
0010+=          CSS3    EQU     CSS2    ;SAME CHECKSUM VECTOR SIZE
EAA1+=          XLT3    EQU     XLT2    ;SAME TRANSLATE TABLE
                        DISKDEF 4,0,14,0,1024,180,64,64,0 ; RAM FLOPPY 60K,120K,180K
EABB+=          DPB4    EQU     $               ;DISK PARM BLOCK
EABB+0F00               DW      15              ;SEC PER TRACK
EABD+03                 DB      3               ;BLOCK SHIFT
EABE+07                 DB      7               ;BLOCK MASK
EABF+00                 DB      0               ;EXTNT MASK
EAC0+B300               DW      179             ;DISK SIZE-1
EAC2+3F00               DW      63              ;DIRECTORY MAX
EAC4+C0                 DB      192             ;ALLOC0
EAC5+00                 DB      0               ;ALLOC1
EAC6+1000               DW      16              ;CHECK SIZE
EAC8+0000               DW      0               ;OFFSET
0000+=          XLT4    EQU     0               ;NO XLATE TABLE

                ; ENDEF AM SCHLUSS NICHT VERGESSEN

F01E =          MON80   EQU 0F01EH      ; NEUSTART DES MONITORS
F01E =          RMON80  EQU 0F01EH
000D =          CR      EQU 0DH         ; ZEICHENDEFINITIONEN
000A =          LF      EQU 0AH

                SIGNON:                 ; MELDUNG NACH DEM KALTSTART
EACA 1A                 DB 26
EACB 57656C636F         DB 'Welcome to 60 K CP/M 2.2'
EAE3 0D0A00             DB CR,LF,0

                CONST:                  ; CONSOL-STATUS
EAE6 C312F0             JMP 0F012H
                CONIN:
EAE9 CD03F0             CALL 0F003H     ; CONSOL-EINGABE
EAEC E67F               ANI 7FH         ; ACHTUNG PARITAET=0
EAEE C9                 RET
                CONOUT:                 ; CONSOL-AUSGABE
EAEF C309F0             JMP 0F009H

F00F =          LIST    EQU 0F00FH      ; DRUCKER
F00C =          PUNCH   EQU 0F00CH      ; PO
F006 =          READER  EQU 0F006H      ; RI

                BOOT:                   ; KALTSTART FOLGT HIER.
EAF2 310001             LXI SP,BUFF+80H ; STACK VORBELEGEN
EAF5 21CAEA             LXI H,SIGNON    ; MELDUNG AUSGEBEN
EAF8 CDF4EB             CALL PRMSG      ; MIT DRUCKROUTINE
EAFB AF                 XRA A           ; LAUFWERK A WIRD ANGEWAEHLT
EAFC 320400             STA CDISK       ;
                ;
                ; SEKTORENBUFFER IST LEER, MONITOR WIRD DESAKTIVIERT
                ;
EAFF AF                 XRA A           ; KEIN SCHREIBVORGANG MEHR AKTUELL
EB00 32F4ED             STA MWRTFLG     ; DAHER AUF 0 SETZEN
EB03 3EFF               MVI A,0FFH      ; LAUFWERK IST UNDEFINIERT
EB05 32F5ED             STA MDRVAKT     ; NACH DEM BOOTEN
EB08 2114EB             LXI H,WBOOT     ; MONITOREINSPRUNG WIRD
EB0B 2234F0             SHLD 0F033H+1   ; KURZGESCHLOSSEN
EB0E 2237F0             SHLD 0F036H+1   ; DENN EVTL. UEBERSCHREIBEN
                ;
EB11 C36DEB             JMP GOCPM       ; UND CP/M DANN STARTEN

                WBOOT:                  ; WARM-BOOT
EB14 3AF4ED             LDA MWRTFLG     ; WENN NOCH EIN ALTER TRACK ZUM
EB17 B7                 ORA A           ; SCHREIBEN DA, DANN ZURUECK DAMIT.
EB18 CA1EEB             JZ NOTBAC       ; SONST WEITER.
EB1B CDC9ED             CALL PUTTRK     ; NORMALERWEISE IST SCHREIBVORGANG
                                        ; NACH EINEM DIREKTORYZUGRIFF ABGESCHLOSSEN
                                        ;
EB1E 3EFF       NOTBAC: MVI A,0FFH      ; ALLE TRACKS UNGUELTIG, BEI DISKETTENWECHSEL
EB20 32F5ED             STA MDRVAKT     ; WICHTIG.

EB23 318000             LXI SP,BUFF     ; STACK ZUWEISEN
EB26 0E05               MVI C,RETRY     ; ANZAHL DER VERSUCHE
EB28 C5                 PUSH B          ; UND DANN ANFANGEN ZU BOOTEN
EB29 0100D4     WBOOT0: LXI B,CPMB      ; BOOT VON MINI-DISKETTE
EB2C CDEEEB             CALL SETDMA     ; AUF DER STARTADRESSE DES CP/MS
EB2F 0E00               MVI C,0         ; LAUFWERK A
EB31 CDB7EB             CALL SELDSK     ; AUSWAEHLEN
EB34 0E00               MVI C,0         ; TRACK 0
EB36 CDCDEB             CALL SETTRK     ; UND DEN ZWEITEN PHYS. SEKTOR, (NR8 LOGISCH)
EB39 0E08               MVI C,8         ; ENSTPRICHT NR 2 BEI 1024 BYTES
EB3B CDD2EB             CALL SETSEC     ; WICHTIG, DA ANDERE ZAEHLWEISE
EB3E C1                 POP B           ; UND VON DA AN N SEKTOREN EINLESEN
EB3F 062C               MVI B,NSECTS    ; ABER DAS BIOS NICHT UEBERSCHREIBEN
                                        ; DAMIT PATCHES LEICHT MOEGLICH SIND
EB41 C5         RDSEC:  PUSH B          ; ANZAHL MERKEN
EB42 CD01EC             CALL READ       ; LESEN AUSFUEHREN
EB45 C297EB             JNZ BOOTERR     ; FEHLER: DEFEKTER SEKTOR
EB48 2AFEED             LHLD IDD        ; ZIELADRESSE LADEN
EB4B 118000             LXI D,128       ; UM LOGISCHE SEKTORGROESSER ERHOEHEN
EB4E 19                 DAD D           ; DAZU ADDIEREN,
```

```
EB4F 22FEED             SHLD IOD        ; UND DANN WIEDER ZURUECKSPEICHERN.
EB52 3AFDED             LDA IOS         ; SEKTOR LADEN
EB55 FE27               CPI 39          ; MINI 0..39 SEKTOREN A 128 BYTES
EB57 DA64EB             JC RD1          ; SOLANGE AUF DER GLEICHEN SPUR BLEIBEN
EB5A 3AFCED             LDA IOT         ; DANN NEUE SPUR ANWAEHLEN,
EB5D 3C                 INR A           ; JEDOCH IM VERFAHREN 0,2,...
EB5E 3C                 INR A           ; SPUR 0, DANN SPUR 2, WEGEN BOOT.ASM
EB5F 32FCED             STA IOT         ; DENN 1,3,5 IST DIE RUECKSEITE DES LAUFWERKS
EB62 3EFF               MVI A,0FFH      ; 0,1,2,3.... NACH INCREMENT A=0
EB64 3C         RD1:    INR A           ; DANN 0
EB65 32FDED             STA IOS         ; UND AUCH NEUESN SEKTOR ANWAEHLEN
EB68 C1                 POP B           ; SCHLEIFENZAEHLER ZURUECH
EB69 05                 DCR B           ; UND IMMER WEITER LESEN
EB6A C241EB             JNZ RDSEC       ; DANACH CP/M NEU STARTEN
                GOCPM:
EB6D 018000             LXI B,BUFF      ; BUFFER AUF DEFAULT EINSTELLEN
EB70 CDEEEB             CALL SETDMA     ; SO WIE ES CP/M BRAUCHT
EB73 3EC3               MVI A,JMP       ; SPRUNG AUF DEN WARM-BOOT IM
EB75 320000             STA 0           ; RAM ABLEGEN.
EB78 2103EA             LXI H,WBOOTE    ; WARM-BOOT-ADRESSE
EB7B 220100             SHLD 1          ; NICHT VERGESSEN
EB7E 320500             STA 5           ; SPRUNG AUF DIE BDOS-CALL-ADRESSE
EB81 2106DC             LXI H,BDOS      ; LEGEN UND AUCH DAS ZIEL
EB84 220600             SHLD 6          ; DORTHIN
EB87 323800             STA 7*8         ; RST7 DEFINIEREN, DEFAULT IST MONITOR
EB8A 211EF0             LXI H,MON80     ; DER ABER NORMALERWEISE KURZGESCHL. IST.
EB8D 223900             SHLD 7*8+1
EB90 3A0400             LDA CDISK       ; DAS ZULETZT VERWENDETE LAUFWERK
EB93 4F                 MOV C,A         ; LADEN UND DAMIT SELEKTIEREN.
EB94 C300D4             JMP CPMB

                BOOTERR:                ; IM FEHLERFALLE, BEI BAD-SEKTOR.
EB97 C1                 POP B           ; ERST MAL NOCHEINMAL VERSUCHEN
EB98 0D                 DCR C           ;
EB99 CAA0EB             JZ BOOTER0      ; BIS HOFFNUNGSLOS, DANN FEHLERMELDUNG
EB9C C5                 PUSH B
EB9D C329EB             JMP WBOOT0      ; TRY AGAIN

                BOOTER0:                ; FEHLERMELDUNG SCHLIESSLICH AUSGEBEN
EBA0 21A9EB             LXI H,BOOTMSG   ; UND MONITOR NEU STARTEN, BZW. WBOOT.
EBA3 CDF4EB             CALL PRMSG
EBA6 C31EF0             JMP MON80

                BOOTMSG:                ; FEHLERMELDUNG
EBA9 3F424F4F54         DB '?BOOT',0

                LISTST:                 ; LST-STATUS, DERZEIT KURZGESCHLOSSEN
EBAF 00                 NOP             ;
EBB0 AF                 XRA A           ; GGF. HIER SPRUNG EINBAUEN.
EBB1 C9                 RET

                HOME:                   ; LAUFWERK, SPUR 0 ANFAHREN
EBB2 0E00               MVI C,0         ; ABER NUR ANWAEHLEN, NICHT
EBB4 C3CDEB             JMP SETTRK      ; AUSFUEHREN

                SELDSK:                 ; LAUFWERK AUSWAEHLEN
EBB7 210000             LXI H,0         ; UND PRUEFEN OB GUELTIG
EBBA 79                 MOV A,C         ; WENN GROESSER ALS NDISKS
EBBB FE05               CPI NDISKS      ; DANN NICHT OK
EBBD D0                 RNC             ; NUMMER 0 BIS N-1 ERSCHEINT IN A
EBBE 32F9ED             STA DBANK       ; DANN ZUSAETZLICH DIE
EBC1 69                 MOV L,C         ; LAUFWERKSTABELLE AUSRECHNEN
EBC2 2600               MVI H,0         ; DAZU NUMMER MIT 16 MULTIPLIZIEREN
EBC4 29                 DAD H           ;
EBC5 29                 DAD H           ;
EBC6 29                 DAD H           ;
EBC7 29                 DAD H           ;
EBC8 1133EA             LXI D,DPBASE    ; UND BASISADRESSE DRAUF ADDIEREN.
EBCB 19                 DAD D
EBCC C9                 RET
                ;
                SETTRK:                 ; SPUR MERKEN
EBCD 21FCED             LXI H,IOT       ; DAZU IM SPEICHERZELLE LADEN
EBD0 71                 MOV M,C
EBD1 C9                 RET

                SETSEC:                 ; SEKTOR MERKEN
EBD2 21FDED             LXI H,IOS       ; DAZU IN SPEICHERZELLE LADEN
EBD5 71                 MOV M,C
EBD6 C9                 RET

                SECTRAN:                ; SEKTOR UMSETZUNG
EBD7 7A                 MOV A,D         ; WENN EIN SKEW-FAKTOR VERWENDET WIRD,
EBD8 B3                 ORA E           ; WIE Z.B. BEI 8 ZOLL UEBLICH.
EBD9 CAE6EB             JZ SE1          ; =0, DANN KEIN SKEW VERWENDET,
EBDC 0600               MVI B,0         ; SONST IN DE ADRESSE DER SKEW-TABELLE
EBDE EB                 XCHG            ; DAZU SEKTOR IN C ADDIEREN
EBDF 09                 DAD B           ; UND WERT ALS NEUEN SEKTOR
EBE0 7E                 MOV A,M         ; FESTLEGEN UND
EBE1 32FDED             STA IOS         ; SPEICHERN.
EBE4 6F                 MOV L,A
EBE5 C9                 RET
EBE6 69         SE1:    MOV L,C         ; SONST NUR EINFACHE WERT
EBE7 79                 MOV A,C         ; UEBERNEHMEN, OHNE UMRECHNUNG.
EBE8 32FDED             STA IOS         ; AUCH MERKEN
EBEB 2600               MVI H,0         ; SE 0..255 MAX
EBED C9                 RET
                ;
                SETDMA:                 ; ADRESSE FUER FLOPPY-ZUGRIFF FESTLEGEN.
EBEE 69                 MOV L,C
EBEF 60                 MOV H,B
EBF0 22FEED             SHLD IOD
EBF3 C9                 RET

                PRMSG:                  ; TEXT AUSGEBEN, FUER FEHLERMELDUNG
EBF4 7E                 MOV A,M         ; DAZU LADEN
EBF5 B7                 ORA A           ; =0, DANN ENDE DES TEXTES
EBF6 C8                 RZ              ; SONST UEBER CONSOLE AUSGEBEN
EBF7 E5                 PUSH H
EBF8 4F                 MOV C,A
EBF9 CDEFEA             CALL CONOUT
EBFC E1                 POP H
EBFD 23                 INX H           ; BIS ALLE BUCHSTABEN DRAUSSEN
EBFE C3F4EB             JMP PRMSG

                ;READ UND WRITE UNTER VERWENDUNG VON EXEC IM MONITOR
                ; HL=DMA ADR
                ; DE=TRACK/SECTOR
                ; B=0 RSTORE
```

```
                ;   1 READ
                ;   2 WRITE
                ; C=DRIVE 0...3         ; BEI MEXEC, EXEC
                ; BEI FEXEC IST C BEI BESTIMMT.

                READ:                   ; EINEN SEKTOR LESEN
EC01 3AF9ED             LDA DBANK       ; DAZU LAUFWERK BESTIMMEN
EC04 FE02               CPI 2           ;
EC06 DADCEC             JC MINIRD       ; 0,1 SIND MINILAUFWERKE
EC09 FE04               CPI 4
EC0B DA57EC             JC MAXIREAD     ; 8 ZOLL LAUFWERK
                ;
                ; RAMFLOPPY ZUSATZ-ROUTINEN
                ;
EC0E CD23EC             CALL ADRERZ     ; HL=QUELLADRESSE
EC11 EB                 XCHG            ; ADRESSUMRECHNUNG DURCHFUEHREN
EC12 2AFEED             LHLD IOD        ; ZIELADRESSE LADEN
EC15 EB                 XCHG            ; UND DE=ZIEL, HL=QUELLE, C=BANK QUELLE
EC16 0600               MVI B,0         ; ZIEL IST BANK 0
EC18 CDF1ED             CALL REXEC      ; UND 128 BYTES KOPIEREN, CARRY=FEHLER
EC1B D221EC             JNC NORERR
EC1E 3E01               MVI A,1         ; FEHLER DA, BANK NICHT VORHANDEN,
EC20 C9                 RET             ; WIRKT WIE BAD-SEKTOR
EC21 AF         NORERR: XRA A           ; SONST OK.
EC22 C9                 RET

                ; SEKTOR 0..E,  TRACK 0..5FH
                ; SSSSTTTT T0000000   , ADRESSE FUER RAM-FLOPPY
                ADRERZ:                 ; ADRESSE BERECHNEN, QUELLE IN HL
EC23 3AFCED             LDA IOT         ; TRACK HOLEN
EC26 0F                 RRC             ; UND UMRECHNEN
EC27 E60F               ANI 0FH         ; UNTERER TEIL VOM MSB
EC29 67                 MOV H,A
EC2A 3AFDED             LDA IOS         ; DANN SEKTOR DAZU
EC2D 07                 RLC
EC2E 07                 RLC
EC2F 07                 RLC
EC30 07                 RLC
EC31 E6F0               ANI 0F0H
EC33 B4                 ORA H
EC34 67                 MOV H,A         ; DAMIT SSSSTTTT OK
EC35 3AFCED             LDA IOT
EC38 0F                 RRC             ; TXXXXXXX
EC39 E680               ANI 80H
EC3B 6F                 MOV L,A         ; T0000000
EC3C 3AFCED             LDA IOT         ; NUN NOCH BANK BESTIMMEN, IN C UND B
EC3F 07                 RLC             ; UND DAZU MSB-TEIL DES TRACKS VERWENDEN
EC40 07                 RLC             ; 0MMTTTTT
EC41 07                 RLC             ; BANKNUMMER
EC42 E603               ANI 03H         ; 000000MM
EC44 C601               ADI 1           ; ERST AB BANK 1 STARTEN
EC46 47                 MOV B,A         ; DA BANK 0=CP/M RAM UND TPA
EC47 4F                 MOV C,A         ; OK BEIDE DEFINIERT
EC48 C9                 RET

                NEUBANK:                ; UMRECHNEN FUER 8ZOLL
                                        ; UND NEUEN FLOPPY-EINSPRUNG VERWENDEN.
                                        ; NACH C LADEN
                                        ; LW=2, DANN VORDERSEITE LW 3
                                        ; LW=3, DANN RUECKSEITE LW 3
EC49 3AF9ED             LDA DBANK       ;
EC4C FE02               CPI 2
EC4E C254EC             JNZ NEU1
EC51 0E14               MVI C,00010100B ; SD,8 ZOLL, LW=3
EC53 C9                 RET
EC54 0E94       NEU1:   MVI C,10010100B
EC56 C9                 RET             ;

                MAXIREAD:               ; LESEN DER 8 ZOLL FLOPPY
EC57 0605       SK1:    MVI B,RETRY     ; ANZAHL DER LESEVERSUCHE
EC59 C5         LP:     PUSH B          ; DANN AUSFUEHREN
EC5A 2AFEED             LHLD IOD        ; ZIELADRESSE HOLEN
EC5D 3AFCED             LDA IOT         ; SPURNUMMER
EC60 57                 MOV D,A
EC61 3AFDED             LDA IOS         ; SEKTORNUMMER
EC64 5F                 MOV E,A
EC65 0601               MVI B,1         ; LESE-BEFEHLSCODE
EC67 CD49EC             CALL NEUBANK    ; VORHER LAUFWERKSCODE UMRECHNEN
EC6A CDE8ED             CALL FEXEC      ; UND DANN AUSFUEHREN
EC6D C1                 POP B           ; RETRY-ZAEHLER
EC6E C8                 RZ              ; KEIN FEHLER, DANN OK ZURUECK
EC6F 05                 DCR B           ; SONST NOCHMALS PROBIEREN
EC70 C259EC             JNZ LP
EC73 3E01               MVI A,1         ; BAD SEKTOR
EC75 B7                 ORA A
EC76 C9                 RET
                ;

                WRITE:                  ; SCHREIBEN EINES SEKTORS
EC77 3AF9ED             LDA DBANK       ; DAZU LAUFWERKSCODE LADEN
EC7A FE02               CPI 2           ; UND FLOPPY-TYP BESTIMMEN
EC7C DA35ED             JC MINIWR       ; 0 UND 1 SIND MINILAUFWERKE
EC7F FE04               CPI 4           ; 2 UND 3 MAXILAUFWERKE
EC81 DA96EC             JC MAXIWR       ; REST IST RAM-FLOPPY
                ; RAM FLOPPY
EC84 CD23EC             CALL ADRERZ     ; HL=QUELLADRESSE
EC87 EB                 XCHG            ; UMRECHNEN
EC88 2AFEED             LHLD IOD        ; ZIEL IN BANK HL=QUELLE DIESMAL
EC8B 0E00               MVI C,0         ; QUELLE IST BANK 0, B=ZIEL, DE=ZIEL
EC8D CDF1ED             CALL REXEC      ; UND 128 BYTES KOPIEREN, CARRY=FEHLER
EC90 D221EC             JNC NORERR      ; OK BANK WAR DA, SONST
EC93 3E01               MVI A,1         ; FEHLER AUSGEBEN
EC95 C9                 RET

EC96 0605       MAXIWR: MVI B,RETRY     ; SCHREIBEN BEI 8 ZOLL
EC98 C5         LPP:    PUSH B          ; DAZU FEHLERZAEHLER RETTEN
EC99 2AFEED             LHLD IOD        ; QUELLEADRESSE LADEN
EC9C 3AFCED             LDA IOT         ; SPUR
EC9F 57                 MOV D,A
ECA0 3AFDED             LDA IOS         ; SEKTOR
ECA3 5F                 MOV E,A
ECA4 0602               MVI B,2         ; SCHREIB-BEFEHLSCODE
ECA6 CD49EC             CALL NEUBANK    ; VORHER LAUFWERK UMRECHNEN
ECA9 CDE8ED             CALL FEXEC      ; UND AUSFUEHREN
ECAC C1                 POP B
ECAD C8                 RZ              ; KEIN FEHLER, DANN ZURUECK
ECAE 05                 DCR B           ; SONST ERNEUT VERSUCHEN
ECAF C298EC             JNZ LPP
ECB2 3E01               MVI A,1         ; BAD SEKTOR
ECB4 B7                 ORA A
ECB5 C9                 RET
```

```
                ; MINIFLOPPY 80 SPUR, DD, DS
                ;
                ;READ UND WRITE UNTER VERWENDUNG VON MEXEC
                ; HL=DMA ADR
                ; DE=TRACK/SECTOR
                ; B=0 RSTORE
                ;   1 READ
                ;   2 WRITE
                ; C=DRIVE 0...3  10H,11H,12H,13H  DOUBMIN 0D0H,0D1H,0D2H,0D3H
                ;                                          A   C,   B    D
                ; 1K BUFFER IN MONITORGEBIET
                ; WIRD DADURCH TEILWEISE UEBERSCHRIEBEN
                ; BEI WARMBOOT MUSS BUFFER GELEERT WERDEN
                ; DEBLOCK WIRD AUS SICHERHEITSGRUENDEN NICHT VERWENDET
                ;

FC00 =          BUFFER  EQU     0FC00H  ; FREIES GEBIET BIS FFFF NUR MONITORBEFEHLE

                CALC:-                  ;RECHNET DBANK IN PHYS LAUFWERK UM
                                        ;RECHNET IOS IN SEKTORBUFFERNR UM
ECB6 3AFDED             LDA IOS         ;0..39 IST DER BEREICH
ECB9 0F                 RRC             ;X000NNNN 0..15
ECBA 0F                 RRC
ECBB 0F                 RRC             ;0,1,2,3,4
ECBC E607               ANI 00000111B   ; MAX
ECBE 3C                 INR A           ;1,2,3,4,5  STARTSEKTOR DES GEBIETS (1K)
ECBF 5F                 MOV E,A         ;IN E ALS PARAMETER
ECC0 3AF9ED             LDA DBANK       ;DRIVE 0->0  1->2
                ;       -- NUR 0,1 SUI 4 ;UND TRACK UMRECHNEN , LAUFWERK 0,1
ECC3 FE01               CPI 1           ; 0,2,4,6,8 IST VORDERSEITE 1,3,5... RUECKSEITE
ECC5 C2CAEC             JNZ CAL2
ECC8 3E02               MVI A,2
ECCA 4F         CAL2:   MOV C,A         ;DRIVE PHYSIKALISCH
ECCB 3AFCED             LDA IOT
ECCE 0F                 RRC             ; TRACK / 2,  = PHYS TRACK, CARRY=RUECKSEITE
ECCF 57                 MOV D,A         ; TRACK MERKEN
ECD0 D2D7EC             JNC CAL3
ECD3 79                 MOV A,C
ECD4 F601               ORI 1           ; D0 -> D1,  D2 -> D3
ECD6 4F                 MOV C,A
ECD7 7A         CAL3:   MOV A,D
ECD8 E67F               ANI 7FH         ; BEREICH TRACK 0..79 REAL
ECDA 57                 MOV D,A         ;TRACK=D SEKTOR=E DRIVE=C
ECDB C9                 RET

                ;
ECDC CDB6EC     MINIRD: CALL CALC       ; FUER VERGLEICH, LAUFWERKSDATEN UMRECHNEN
ECDF 3AF5ED             LDA MDRVAKT     ; UND NUN AKTUELLES LAUFWERK VERGLEICHEN
ECE2 B9                 CMP C           ; WENN NICHT GLEICH, DANN NEU
ECE3 C210ED             JNZ RLOAD       ; LADEN,
ECE6 3AF6ED             LDA MTRKAKT     ; SONST SPUR VERGLEICHEN
ECE9 BA                 CMP D           ; UND
ECEA C210ED             JNZ RLOAD
ECED 3AF7ED             LDA MSEKAKT     ; SONST SEKTOR
ECF0 BB                 CMP E           ; WENN GLEICH, DANN SEKTOR IM SPEICHER
ECF1 C210ED             JNZ RLOAD       ; UND LADEN UNNOETIG
                R1RD:                   ; OK IST SCHON IN BUFFER
ECF4 2100FC             LXI H,BUFFER    ; ADRESSE BERECHNEN
ECF7 3AFDED             LDA IOS         ; 0..39   * 128 + BUFFER
ECFA E607               ANI 00000111B   ; 0,1,2,3,4,5,6,7
ECFC 57                 MOV D,A
ECFD 1E00               MVI E,0         ; SCHIEBEN MIT Z80 BEFS
ECFF CB2A               DB 0CBH,2AH     ; SRA D
ED01 CB1B               DB 0CBH,1BH     ; RR E   = *256/2
ED03 19                 DAD D           ; +BUFFER
ED04 EB                 XCHG            ; NACH DE IST ZIEL
ED05 2AFEED             LHLD IOD        ; DMA ADRESSE QUELLE HIER
ED08 EB                 XCHG            ; ZIEL DE
ED09 018000             LXI B,128       ; LAENGE
ED0C EDB0               DB 0EDH,0B0H    ; LDIR
ED0E AF                 XRA A
ED0F C9                 RET             ; OK ENDE

                RLOAD:                  ; NEUEN LADEN, GGF ALTEN ZURUECKSCHREIBEN
ED10 3AF4ED             LDA MWRTFLG
ED13 B7                 ORA A
ED14 CA1DED             JZ R1LOAD
ED17 CDC9ED             CALL PUTTRK     ; ALTEN ZURUECKSCHREIBEN
ED1A DAA4ED             JC ERRIO
ED1D CDB6EC     R1LOAD: CALL CALC       ; BERECHNEN
ED20 79                 MOV A,C
ED21 32F5ED             STA MDRVAKT
ED24 7B                 MOV A,E
ED25 32F7ED             STA MSEKAKT
ED28 7A                 MOV A,D
ED29 32F6ED             STA MTRKAKT
ED2C CDA8ED             CALL GETTRK     ; NEUEN SEKTOR LESEN
ED2F DAA4ED             JC ERRIO        ; FEHLER AUFGETRETEN
ED32 C3F4EC             JMP R1RD        ; ENDE
                ;
                MINIWR:                 ; SCHREIBEN EINES SEKTORS
ED35 79                 MOV A,C
ED36 32F8ED             STA ALLOC       ; INFORMATION 1=DIREKTORY WRITE
ED39 CDB6EC             CALL CALC       ; FUER VERGLEICH, BERECHUNG AUSFUEHREN
ED3C 3AF5ED             LDA MDRVAKT     ; UND WIE BEI MINIRD
ED3F B9                 CMP C           ;
ED40 C27FED             JNZ WLOAD
ED43 3AF6ED             LDA MTRKAKT
ED46 BA                 CMP D
ED47 C27FED             JNZ WLOAD
ED4A 3AF7ED             LDA MSEKAKT
ED4D BB                 CMP E
ED4E C27FED             JNZ WLOAD       ; LADEN UNNOETIG, SEKTOR SCHON DA.
                W1WR:                   ; OK IST SCHON IN BUFFER
ED51 2100FC             LXI H,BUFFER    ; ADRESSE BERECHNEN
ED54 3AFDED             LDA IOS         ; 0..39   * 128 + BUFFER
ED57 E607               ANI 00000111B   ; 0,1,2,3,4,5,6,7
ED59 57                 MOV D,A
ED5A 1E00               MVI E,0         ; SCHIEBEN MIT Z80 BEFS
ED5C CB2A               DB 0CBH,2AH     ; SRA D
ED5E CB1B               DB 0CBH,1BH     ; RR E   = *256/2
ED60 19                 DAD D           ; +BUFFER
ED61 EB                 XCHG            ; NACH DE IST ZIEL
ED62 2AFEED             LHLD IOD        ; DMA ADRESSE QUELLE HIER
ED65 018000             LXI B,128       ; LAENGE ZIEL DE
ED68 EDB0               DB 0EDH,0B0H    ; LDIR
ED6A 3E01               MVI A,1
ED6C 32F4ED             STA MWRTFLG     ; NUN BESCHRIEBEN
ED6F 3AF8ED             LDA ALLOC       ; =1 DANN ZURUECK
```

```
ED72 FE01           CPI 1           ; WENN DIREKTORY ZUGRIFF, DANN
ED74 C27DED         JNZ W2WR        ; GLEICH ZURUECKSCHREIBEN.
ED77 CDC9ED         CALL PUTTRK     ;
ED7A DAA4ED         JC ERRIO        ; FALLS FEHLER, DANN BAD SEKTOR
                    W2WR:
ED7D AF             XRA A           ; KEIN FEHLER
ED7E C9             RET             ; OK ENDE

                    WLOAD:          ; NEUEN LADEN, GGF ALTEN ZURUECKSCHREIBEN
ED7F 3AF4ED         LDA MWRTFLG     ;
ED82 B7             ORA A
ED83 CA8CED         JZ W1LOAD
ED86 CDC9ED         CALL PUTTRK     ; ALTEN ZURUECKSCHREIBEN
ED89 DAA4ED         JC ERRIO
ED8C CDB6EC         W1LOAD: CALL CALC   ; BERECHNEN
ED8F 79             MOV A,C
ED90 32F5ED         STA MDRVAKT
ED93 7B             MOV A,E
ED94 32F7ED         STA MSEKAKT
ED97 7A             MOV A,D
ED98 32F6ED         STA MTRKAKT
ED9B CDA8ED         CALL GETTRK
ED9E DAA4ED         JC ERRIO
EDA1 C351ED         JMP W1WR

EDA4 3E01       ERRIO:  MVI A,1     ; FEHLER AUFGETRETEN
EDA6 B7             ORA A
EDA7 C9             RET

                ; BUFFERVERWALTUNG

                GETTRK:             ;TRKAKT,SEKAKT,DRVAKT ENTHALTEN NEUE
                                    ;BUFFERADRESSE
EDA8 AF             XRA A
EDA9 32F4ED         STA MWRTFLG     ; SEKTOR EINLESEN
EDAC 2100FC         LXI H,BUFFER
EDAF 3AF7ED         LDA MSEKAKT
EDB2 5F             MOV E,A
EDB3 3AF5ED         LDA MDRVAKT
EDB6 F6D0           ORI 11010000B   ;LAUFWERK PHYS 0,1,2,3 DOUBLE DENSE
EDB8 4F             MOV C,A
EDB9 0601           MVI B,1         ;READ, 1K DIREKT
EDBB 3AF6ED         LDA MTRKAKT
EDBE 57             MOV D,A
EDBF CDEEED         CALL MEXEC
EDC2 DAC7ED         JC ERRX
EDC5 AF             XRA A
EDC6 C9             RET
                ;

EDC7 37         ERRX:   STC
EDC8 C9             RET


                PUTTRK:             ;TRKAKT,SEKAKT,DRVAKT ENTHALTEN NEUE
                                    ;BUFFERADRESSE
EDC9 AF             XRA A           ;WIRD ZURUECKGESCHRIEBEN
EDCA 32F4ED         STA MWRTFLG
EDCD 2100FC         LXI H,BUFFER
```

```
EDD0 3AF7ED         LDA MSEKAKT
EDD3 5F             MOV E,A
EDD4 3AF5ED         LDA MDRVAKT
EDD7 F6D0           ORI 11010000B   ;LAUFWERK PHYS 0,1,2,3 DOUBLE DENSE
EDD9 4F             MOV C,A
EDDA 0602           MVI B,2         ;WRITE  1K SEKTOR
EDDC 3AF6ED         LDA MTRKAKT
EDDF 57             MOV D,A
EDE0 CDEEED         CALL MEXEC
EDE3 DAC7ED         JC ERRX
EDE6 AF             XRA A
EDE7 C9             RET

EDE8 C321F0     FEXEC:  JMP 0F021H      ;FLOMON NEUER VEKTOR
EDEB C324F0     EXEC:   JMP 0F024H      ;FLOMON UND MC
EDEE C327F0     MEXEC:  JMP 0F027H      ;FLOMON UND MC
EDF1 C35BF0     REXEC:  JMP 0F05BH      ;FLOMON RAM-FLOPPY


                                        ;SOFT SYSTEM
                ;
                ; RAM ZELLEN

EDF4 00         MWRTFLG:        DB 0    ;<>0 IST WRITEN
EDF5 00         MDRVAKT:        DB 0    ;DRIVE DAS GELADEN IST
EDF6 00         MTRKAKT:        DB 0    ;TRACK
EDF7 00         MSEKAKT:        DB 0    ;SEKTOR
EDF8 00         ALLOC:          DB 0    ; MERKER, 1=DIREKTORY WRITE
                ;

                ;
EDF9 00         DBANK:  DB 0
EDFA 80         IOPB:   DB 80H  ;NORM IO
EDFB 01         ION:    DB 1    ;SECTOR NR
EDFC 04         IOT:    DB OFFSET ;TRK
EDFD 01         IOS:    DB 1
EDFE 8000       IOD:    DW BUFF
EE00 01         ALTDRV: DB 1
EE01 0000       INDADR: DW 0
EE03 0000       INDADR2:DW 0


                        ENDEF
EE05+=          BEGDAT  EQU     $
EE05+           DIRBUF: DS      128     ;DIRECTORY ACCESS BUFFER
EE85+           ALV0:   DS      49
EEB6+           CSV0:   DS      64
EEF6+           ALV1:   DS      49
EF27+           CSV1:   DS      64
EF67+           ALV2:   DS      31
EF86+           CSV2:   DS      16
EF96+           ALV3:   DS      31
EFB5+           CSV3:   DS      16
EFC5+           ALV4:   DS      23
EFDC+           CSV4:   DS      16
EFEC+=          ENDDAT  EQU     $
01E7+=          DATSIZ  EQU     $-BEGDAT

EFEC                    END
```

man noch nicht stolzer Besitzer von CP/M ist (dann muß es vor dem Start aber nach 100h verschoben werden, z. B. mit dem Z80-LDIR-Befehl, und davor muß man die Bank umschalten).

Beispiel:

```
Adresse > 8800          ; freien Platz
                          nehmen
          LD A,80h      ; Bank 0 ohne
                          EPROM an-
                          wählen
          OUT (0C8H),A; und schalten
          LD HL,8800h   ; Startadresse
          LD DE,100h    ; Zieladresse
          LD BC,länge   ; Programmlänge
                          eintragen
          LDIR          ; transportieren
          JP 100h       ; und starten
```

Vor dem Start alles sichern, z. B. auf Kassette, solange CP/M nicht läuft. Wenn man das NDR-Format herstellen will, so wählt man im Menü den Punkt 5 aus. Danach erscheint *Bild 9*. Das Programm unterstützt auch die Floppy-Controller des mc-Computers, daher müssen wir hier FLO2, also 1 auswählen. Dann erscheint *Bild 10*. Dort wählt man das Laufwerk aus, hier also A oder B; nimmt man A, muß man 1 eingeben. Die Rückseitenauswahl erfolgt automatisch, und ist im Menü nur für Spezialanwendungen verfügbar. Es erscheint die Kontrollausgabe (*Bild 11*).

Alle wichtigen Laufwerksdaten werden nocheinmal ausgegeben. Wenn man jetzt die Taste „J" zur Bestätigung drückt, beginnt die Formatierung. Achtung, keine Diskette mit Daten einlegen, die werden beim Formatieren unter Garantie gelöscht. Das Formatieren dauert etwa 2 Min. *Bild 12* zeigt den Bildschirm während der Formatierung. Die erste Reihe wird beim eigentlichen Formatieren ausgegeben, jedes F steht für eine Spur, bei uns für Vorder- und Rückseite, das V darunter für einen Prüflesevorgang. Tritt ein Fehler auf, wird eine Meldung auf dem Bildschirm mit Fehlerhinweis ausgegeben.

```
A>sysgen80
SYSGEN VER 2.0
SOURCE DRIVE NAME (OR RETURN TO SKIP)a
SOURCE ON A, THEN TYPE RETURN
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)a
DESTINATION ON A, THEN TYPE RETURN
```

Bild 5. SYSGEN80-Bedienung

```
                ;*************************************************
                ;* COLD - BOOT PROGRAMM FUER MINILAUFWERKE       *
                ;* ROLF-DIETER KLEIN 841204    1.0               *
                ;* 80 SPUR LAUFWERKE   1K PRO SEKTOR BOOT        *
                ;*************************************************

                ; DAS PROGRAMM WIRD AUF SEKTOR 1 TRACK 0 ABGELEGT

FC00                    ORG 0FC00H              ;ANFANGSADRESSE, BEI DD

A000 =          OFFSET  EQU     0A000H          ;
D400 =          CPMB    EQU     03400H+OFFSET   ;START CP/M
EA00 =          BIOS    EQU     04A00H+OFFSET   ;START BIOS

                WBOOT0:

FC00 310001             LXI SP,100H     ;START OF STACK
                ;
FC03 2100D4             LXI H,CPMB      ;START ADRESSE
FC06 0607               MVI B,7         ;ANZAHL DER SEKTOREN  A 1024 BYTES
                                        ; FUELLT GENAU BIS 0EFFF
                                        ;1C00H BYTES (GROESSER ALS STD 34H/8)
FC08 1600               MVI D,0         ;START BEI TRACK 0 SEKTOR 2, DA 1 BOOTSEKTOR
FC0A 1E02               MVI E,2         ;1..5 SEKTOR
                RDSEC:
FC0C E5                 PUSH H
FC0D D5                 PUSH D
FC0E C5                 PUSH B
FC0F 0ED0               MVI C,11010000B ;LAUFWERK 0 DOUBLE DENSE MINI
FC11 0601               MVI B,1         ;READ  BEI SYSGEN 2 SETZEN
FC13 CD27F0             CALL 0F027H
FC16 C1                 POP B
FC17 D1                 POP D
FC18 E1                 POP H
FC19 DA1EF0             JC 0F01EH       ;MONITOR AUFRUFEN BEI FEHLER
FC1C D5                 PUSH D          ;NEXT ADRESSE
FC1D 110004             LXI D,1024      ;DOUBLE DENSE BLOECKE
FC20 19                 DAD D
FC21 D1                 POP D
FC22 1C                 INR E           ;SEKTORANZAHL + 1
FC23 7B                 MOV A,E
FC24 FE06               CPI 5+1         ;1..5  ERST BEI 6 NEUE SPUR (4 MAX)
FC26 DA2CFC             JC RD1
FC29 1E01               MVI E,1         ;START OVER
FC2B 14                 INR D
FC2C 05         RD1:    DCR B
FC2D C20CFC             JNZ RDSEC
FC30 C300EA             JMP BIOS        ;START COLDBIOS

FC33                    END
```

Bild 4. Das BOOT-Programm lädt das System in den Speicher

Wichtige Hinweise

Wenn der Hinweis „CPU zu langsam" auftaucht, hat man entweder einen falschen Quarz in der CPU (es müssen 4 MHz CPU-Takt vorliegen, also 8 MHz Quarz) oder man bearbeitet ein Hoch-Dichte-Format, wie z. B. 8 Zoll mit doppelter Dichte. Dort benötigt man eine 6-MHz-CPU.

Wenn man nicht die fertigen Formate im Menü anwählt, muß man weitere Angaben machen. *Bild 13* zeigt die Auswahl der Spurenzahl, *Bild 14* die Laufwerksart. SS0 bedeutet hier, daß auf der Rückseite das SS0-Bit im Format gesetzt wird (wie es z. B. beim 5¼-Zoll-mc-Format der Fall ist, während 8 Zoll doppelseitig beim mc-Format ohne SS0 formatiert werden muß). *Bild 15* zeigt, wie man die Dichte auswählt. Da gibt es wieder eine Ausnahme, das ECMA-70-Format im Hauptmenü formatiert die erste Spur in einfacher Dichte, den Rest in doppelter Dichte. *Bild 16* zeigt die Sektorauswahl bei einfacher Dichte 5¼ Zoll, *Bild 17* doppelte Dichte bei 5¼ Zoll, *Bild 18* einfache Dichte bei 8 Zoll und *Bild 19* doppelte Dichte bei 8 Zoll, wofür man aber eine mit 6 MHz betriebene CPU benötigt. Übrigens kann man auch Laufwerke mit 3½ Zoll und 3 Zoll betreiben, da die meisten steckerkompatibel sind. Bei diesen Laufwerken sind zwei Seiten mit je 80 Spuren schon neuer Standard geworden. Das mc-BIOS bleibt unverändert.

Bibl.d.TU.

```
                ;************************************
                ;* Stepraten Einstellung, diese     *
                ;* bleibt bis zum naechsten Kalt-   *
                ;* Start erhalten                   *
                ;* Alle Laufwerke sind betroffen    *
                ;* Rolf-Dieter Klein 841220   1.0   *
                ;************************************

                .z80

F027            mini    equ     0f027h

0000'           start:
0000'   06 00           ld b,0  ; Stepraten-Befehl
0002'   16 00           ld d,0  ; 0=maximale Steprate (0..3)
0004'   CD F027         call mini       ; und setzen
0007'   C3 0000         jp 0    ; anschliessend Warm-Start
                        end
```

Bild 6. So wird die Steprate eingestellt

```
rom  abs                                                        checksum
0100  C3 00 07 C3 00 07 C3 09 F0 C3 03 F0 F1 DB 40 DB   += 07ED
0110  30 DB C0 FB ED 4D 21 07 15 CD 75 0B DB 40 DB 30   += 07B0
0120  DB C0 ED 56 3E C3 32 38 00 21 0C 01 22 39 00 CD   += 059F
0130  8E 04 AF 32 61 18 3A 5B 18 47 C5 CD F3 01 CD E3   += 0716
0140  02 CD 66 06 CD 45 02 3A 5C 18 FE 01 C2 78 01 3A   += 0571
0150  5E 18 F6 80 32 5E 18 3A 60 18 B7 CA 63 01 3E 80   += 05E9
0160  32 5F 18 CD E3 02 CD 66 06 CD 45 02 3A 5E 18 E6   += 063E
0170  7F 32 5E 18 AF 32 5F 18 3A 61 18 3C 32 61 18 47   += 0460
0180  3A 5B 18 B8 CA 8A 01 CD 46 05 0E 46 CD 06 01 C1   += 05BB
0190  10 A8 21 D3 17 CD 75 0B CD 8E 04 AF 32 61 18 3A   += 0603
01A0  5B 18 47 C5 CD F3 01 CD 62 04 3A 5C 18 FE 01 C2   += 06E2
01B0  D5 01 3A 5E 18 F6 80 32 5E 18 3A 60 18 B7 CA C6   += 069D
01C0  01 3E 80 32 5F 18 CD 62 04 3A 5E 18 E6 7F 32 5E   += 0540
01D0  18 AF 32 5F 18 3A 61 18 3C 32 61 18 47 3A 5B 18   += 03FE
01E0  B8 CA E7 01 CD 46 05 0E 56 CD 06 01 C1 10 B4 CD   += 070C
01F0  8E 04 C9 3A 55 18 B7 C8 AF 32 54 18 3A 61 18 B7   += 0638
0200  C2 25 02 21 80 00 22 56 18 3E 00 32 59 18 3E 00   += 0339
0210  32 5C 18 3E 10 32 5D 18 3E 1B 32 5A 18 3E 28 32   += 0330
0220  5B 18 C3 44 02 21 00 01 22 56 18 3E 01 32 59 18   += 0310
0230  3E 00 32 5C 18 3E 10 32 5D 18 3E 36 32 5A 18 3E   += 032F
0240  28 32 5B 18 C9 B7 C8 F5 21 BC 16 CD 75 0B 3A 61   += 06E5
0250  18 6F 26 00 CD AE 02 21 C7 16 CD 75 0B 3A 62 18   += 0529
0260  6F 26 00 CD AE 02 CD 1F 0D F1 CB 57 21 88 16 CA   += 06A7
0270  75 02 21 16 16 CB 5F CA 7D 02 21 DE 15 CB 67 CA   += 0647
0280  85 02 21 A7 15 CB 6F CA 8D 02 21 70 15 CB 77 CA   += 06A9
0290  95 02 21 39 15 CA 9B 02 21 39 15 CD 75 0B CD 09   += 04FF
02A0  01 FE 57 CA AB 02 FE 77 C2 9E 02 C3 00 07 AF F5   += 0812
02B0  3E 10 01 0A 00 11 00 00 EB F5 CB 23 CB 12 CB 15   += 04F5
02C0  CB 14 CB C3 AF ED 42 D2 CD 02 09 CB 83 F1 3D 20   += 0891
02D0  E8 EB 7B F6 30 F5 7C B5 20 D6 F1 B7 C8 4F CD 06   += 0A22
02E0  01 18 F7 DD 21 63 18 3A 59 18 B7 C2 F5 02 06 10   += 05BA
02F0  0E FF C3 F9 02 06 20 0E 4E DD 71 00 DD 23 05 C2   += 0662
0300  F9 02 3E 01 32 62 18 3A 5D 18 47 C5 3A 59 18 B7   += 0503
0310  C2 82 03 06 06 DD 36 00 00 DD 23 05 C2 15 03 DD   += 0522
0320  36 00 FE DD 23 3A 61 18 DD 77 00 DD 23 3A 5F 18   += 05EC
0330  E6 80 B7 C2 3D 03 DD 36 00 00 C3 41 03 DD 36 00   += 064C
0340  01 DD 23 3A 62 18 DD 77 00 DD 23 2A 56 18 7C FE   += 061B
0350  04 C2 56 03 3E 03 DD 77 00 DD 23 DD 36 00 F7 DD   += 069B
0360  23 06 0B DD 36 00 FF DD 23 05 C2 63 03 06 06 DD   += 055C
0370  36 00 00 DD 23 05 C2 6F 03 DD 36 00 FB DD 23 C3   += 0640
0380  06 04 06 0C DD 36 00 00 DD 23 05 C2 84 03 06 03   += 0386
0390  DD 36 00 F5 DD 23 05 C2 90 03 DD 36 00 FE DD 23   += 0773
03A0  3A 61 18 DD 77 00 DD 23 3A 5F 18 E6 80 B7 C2 B8   += 074F
03B0  03 DD 36 00 00 C3 BC 03 DD 36 00 01 DD 23 3A 62   += 0548
03C0  18 DD 77 00 DD 23 2A 56 18 7C FE 04 C2 D1 03 3E   += 0656
03D0  03 DD 77 00 DD 23 DD 36 00 F7 DD 23 06 16 DD 36   += 0690
03E0  00 4E DD 23 05 C2 DE 03 06 0C DD 36 00 00 DD 23   += 051B
03F0  05 C2 EA 03 06 03 DD 36 00 F5 DD 23 05 C2 F6 03   += 0685
0400  DD 36 00 FB DD 23 3A 58 18 4F 2A 56 18 DD 71 00   += 05ED
rom  abs                                                        checksum
0410  DD 23 2B 7C B5 C2 0D 04 DD 36 00 F7 DD 23 3A 59   += 06CC
0420  18 B7 C2 2A 04 0E FF C3 2C 04 0E 4E 3A 5A 18 47   += 050E
0430  DD 71 00 DD 23 05 C2 30 04 3A 62 18 3C 32 62 18   += 04E5
0440  C1 05 C2 0B 03 3A 59 18 B7 C2 51 04 0E FF C3 53   += 0632
0450  04 0E 4E 21 E8 03 DD 71 00 DD 23 2B 7C B5 C2 56   += 062E
0460  04 C9 CD 82 04 3E 01 32 62 18 3A 5D 18 47 C5 CD   += 0593
0470  76 05 CD 45 02 3A 62 18 3C 32 62 18 C1 05 C2 6E   += 0521
0480  04 C9 C5 01 05 0D 0B 79 B0 C2 86 04 C1 C9 3A 53   += 063C
0490  18 B7 C2 C6 04 21 FA 00 E5 DB C0 CD 82 04 FB CD   += 0911
04A0  2E 05 F3 E1 DB C0 E6 80 CA 2D 05 2B 7C B5 20 E8   += 0868
04B0  21 D6 16 CD 75 0B CD 09 01 FE 57 CA C3 04 FE 77   += 078C
04C0  C2 B6 04 C3 00 07 FE 01 C2 FC 04 21 FA 00 E5 DB   += 07E2
04D0  40 CD 82 04 FB CD 36 05 F3 E1 DB 40 E6 80 CA 2D   += 08E2
04E0  05 2B 7C B5 20 E8 21 D6 16 CD 75 0B CD 09 01 FE   += 0698
04F0  57 CA F9 04 FE 77 C2 EC 04 C3 00 07 21 FA 00 E5   += 080F
0500  DB 30 CD 82 04 FB CD 3E 05 F3 E1 DB 30 E6 80 CA   += 0978
0510  2D 05 2B 7C B5 20 E8 21 D6 16 CD 75 0B CD 09 01   += 05C7
0520  FE 57 CA 2A 05 FE 77 C2 1D 05 C3 00 07 C9 CD 23   += 072A
0530  06 3E 0B D3 C0 76 CD 23 06 3E 0B D3 40 76 CD 23   += 0610
0540  06 3E 0B D3 30 76 CD 82 04 FB CD 4F 05 F3 C9 CD   += 07C0
0550  23 06 3A 53 18 B7 C2 60 05 3E 5B D3 C0 C3 75 05   += 0615
0560  FE 01 C2 6C 05 3E 5B D3 40 C3 75 05 FE 02 C2 75   += 0752
0570  05 3E 5B D3 30 76 06 02 C5 FB CD A2 05 F3 C1 3A   += 0741
0580  53 18 B7 C2 8B 05 DB C0 C3 9C 05 FE 01 C2 95 05   += 07CE
0590  DB 40 C3 9C 05 FE 02 C2 9C 05 DB 30 E6 3C C8 10   += 07E7
05A0  D7 C9 CD 23 06 3A 53 18 B7 C2 D1 05 3A 62 18 D3   += 0711
05B0  C2 21 63 18 0E C3 3A 5F 18 E6 80 07 07 E6 02 C6   += 0602
05C0  88 D3 C0 DB C4 07 D2 C3 05 ED A2 C3 C3 05 C3 22   += 095A
05D0  06 FE 01 C2 FB 05 3A 62 18 D3 42 21 63 18 0E 43   += 057D
05E0  3A 5F 18 E6 80 07 07 E6 02 C6 88 D3 40 DB 44 07   += 0694
05F0  D2 ED 05 ED A2 C3 ED 05 C3 22 06 FE 02 C2 22 06   += 07DD
0600  3A 62 18 D3 32 21 63 18 0E 33 3A 5F 18 E6 80 07   += 04B4
0610  07 E6 02 C6 88 D3 30 DB 34 07 D2 17 06 ED A2 C3   += 0797
0620  17 06 C9 C5 3A 5E 18 47 3A 59 18 B7 C2 33 06 78   += 0577
0630  F6 10 47 3A 54 18 B7 C2 3E 06 78 F6 20 47 3A 53   += 0612
0640  18 B7 C2 4C 06 78 C1 D3 C4 C3 65 06 FE 01 C2 5A   += 07FC
0650  06 78 C1 EE 10 D3 44 C3 65 06 FE 02 C2 65 06 78   += 0727
0660  C1 EE 10 D3 34 C9 CD 82 04 FB CD 8E 06 F3 3A 53   += 08BE
0670  18 B7 C2 7A 06 DB C0 C3 8B 06 FE 01 C2 84 06 DB   += 0826
0680  40 C3 8B 06 FE 02 C2 8B 06 DB 30 E6 44 C9 CD 23   += 07D5
0690  06 3A 53 18 B7 C2 B8 06 21 63 18 0E C3 3A 5F 18   += 0500
06A0  E6 80 07 07 E6 02 F6 F4 D3 C0 DB C4 07 D2 AA 06   += 0901
06B0  ED A3 C3 AA 06 C3 FF 06 FE 01 C2 DD 06 21 63 18   += 080B
06C0  0E 43 3A 5F 18 E6 80 07 07 E6 02 F6 F4 D3 40 DB   += 0736
06D0  44 07 D2 CF 06 ED A3 C3 CF 06 C3 FF 06 FE 02 C2   += 08A4
06E0  FF 06 21 63 18 0E 33 3A 5F 18 E6 80 07 07 E6 02   += 04EF
06F0  F6 F4 D3 30 DB 34 07 D2 F4 06 ED A3 C3 F4 06 C9   += 09E5
0700  21 CA 0E CD 75 0B CD 09 01 FE 37 D2 06 07 FE 31   += 0660
0710  DA 06 07 F5 AF 32 55 18 32 53 18 3E E5 32 58 18   += 058C
rom  abs                                                        checksum
0720  AF 32 5F 18 32 60 18 F1 FE 31 C2 40 07 AF 32 54   += 0660
0730  18 CD 0F 09 CD 55 09 CD 90 09 CD B3 09 C3 F1 07   += 06D2
0740  FE 32 C2 59 07 3E 01 32 54 18 CD 0F 09 CD 55 09   += 053F
0750  CD 90 09 CD B3 09 C3 F1 07 FE 33 C2 85 07 3E 01   += 0768
0760  32 54 18 21 80 00 22 56 18 3E 00 32 59 18 3E 1B   += 0309
0770  32 5A 18 3E 00 32 5C 18 3E 1A 32 5D 18 3E 4D 32   += 0344
0780  5B 18 C3 F1 07 FE 34 C2 B8 07 AF 32 54 18 3E 01   += 066D
0790  32 55 18 32 60 18 21 00 01 22 56 18 3E 01 32 59   += 02C5
07A0  18 3E 00 32 5C 18 3E 10 32 5D 18 3E 36 32 5A 18   += 0309
07B0  3E 28 32 5B 18 C3 F1 07 FE 35 C2 E9 07 3E 00 32   += 061B
07C0  54 18 21 00 04 22 56 18 3E 01 32 59 18 3E 36 32   += 02A9
07D0  5A 18 3E 05 32 5D 18 3E 01 32 5C 18 3E 01 32 60   += 0312
07E0  18 3E 50 32 5B 18 C3 F1 07 FE 36 C2 F1 07 C3 FB   += 07B2
07F0  08 21 B0 14 CD 75 0B CD 09 01 FE 34 30 05 FE 31   += 05A7
0800  D2 08 08 FE 0D C2 F7 07 FE 32 C2 12 08 3E 01 C3   += 06BB
0810  1D 08 FE 33 C2 1C 08 3E 02 C3 1D 08 AF 32 53 18   += 04B0
0820  21 A0 13 CD 75 0B CD 09 01 FE 3A D2 26 08 FE 30   += 065E
0830  DA 26 08 FE 39 D2 A2 08 FE 31 DA A2 08 FE 31 C2   += 085F
0840  47 08 3E 01 C3 8A 08 FE 32 C2 51 08 3E 02 C3 8A   += 05BB
0850  08 FE 33 C2 5B 08 3E 04 C3 8A 08 FE 34 C2 65 08   += 0656
0860  3E 08 C3 8A 08 FE 35 C2 6F 08 3E 81 C3 8A 08 FE   += 0719
0870  36 C2 79 08 3E 82 C3 8A 08 FE 37 C2 83 08 3E 84   += 06D2
0880  C3 8A 08 FE 38 C2 8A 08 3E 88 32 5E 18 3A 60 18   += 05FF
0890  B7 CA 9F 08 3A 5E 18 E6 80 CA 9F 08 32 5F 18 C3   += 071B
08A0  BB 08 FE 39 C2 AD 08 C3 00 07 C3 BB 08 FE 30 C2   += 07B1
08B0  B8 0B C3 FB 0B C3 BB 08 C3 20 08 CD 84 0B 21 98   += 070C
08C0  0E CD 75 0B CD 09 01 FE 4A CA E0 08 FE 6A CA E0   += 083E
08D0  08 FE 6E CA E0 08 FE 4E CA E0 08 FE 03 C2 BB 08   += 08AA
08E0  FE 6E CA EA 08 FE 4E C2 ED 08 C3 00 07 FE 03 C2   += 08B8
08F0  F5 08 C3 FB 08 CD 16 01 C3 20 08 21 01 18 CD 75   += 060E
0900  0B CD 09 01 FE 0D C2 01 09 ED 46 CD 00 00 C9 21   += 05A3
0910  64 10 CD 75 0B CD 09 01 FE 36 D2 15 09 FE 31 DA   += 06C5
0920  15 09 FE 31 C2 2C 09 3E 23 C3 51 09 FE 32 C2 36   += 05EA
0930  09 3E 28 C3 51 09 FE 33 C2 40 09 3E 46 C3 51 09   += 0569
0940  FE 34 C2 4A 09 3E 4D C3 51 09 FE 35 C2 51 09 3E   += 067C
0950  50 32 5B 18 C9 21 BA 10 CD 75 0B CD 09 01 FE 34   += 05FF
0960  D2 5B 09 FE 31 DA 5B 09 FE 31 C2 75 09 3E 00 32   += 0682
0970  5C 18 C3 8C 09 FE 32 C2 82 09 3E 01 32 5C 18 C3   += 05F1
0980  8C 09 3E 01 32 5C 18 3E 01 32 60 18 32 5C 18 C9   += 03D2
0990  21 26 10 CD 75 0B CD 09 01 FE 33 D2 96 09 FE 31   += 064C
09A0  DA 96 09 FE 31 C2 AD 09 3E 00 C3 AF 09 3E 01 32   += 064A
09B0  59 1B C9 3A 54 18 B7 C2 8F 0A 3A 59 18 B7 C2 1C   += 0632
09C0  0A 21 1A 11 CD 75 0B CD 09 01 FE 34 D2 C7 09 FE   += 064C
09D0  31 DA C7 09 FE 31 C2 EC 09 21 80 00 22 56 18 3E   += 0630
09E0  10 32 5D 18 3E 1B 32 5A 18 C3 19 0A FE 32 C2 04   += 0490
```

Bild 7. Ein Universal-Formatierer. Er wendet sich direkt an das BIOS

```
09F0  0A 21 B0 00 22 56 18 3E 12 32 5D 18 3E 0A 32 5A   += 0306
0A00  18 C3 19 0A FE 33 C2 19 0A 21 00 01 22 56 18 3E   += 0404
0A10  0A 32 5D 18 3E 10 32 5A 18 C3 8C 0A 21 9E 11 CD   += 0499
0A20  75 0B CD 09 01 FE 35 D2 22 0A FE 31 DA 22 0A FE   += 06BB
rom  abs                                          checksum
0A30  31 C2 47 0A 21 00 01 22 56 18 3E 10 32 5D 18 3E   += 0329
0A40  36 32 5A 18 C3 8C 0A FE 32 C2 5F 0A 21 00 02 22   += 04D3
0A50  56 18 3E 09 32 5D 18 3E 36 32 5A 18 C3 8C 0A FE   += 04CB
0A60  33 C2 77 0A 21 00 02 22 56 18 3E 0A 32 5D 18 3E   += 0356
0A70  28 32 5A 18 C3 8C 0A FE 34 C2 8C 0A 21 00 04 22   += 04F6
0A80  56 18 3E 05 32 5D 18 3E 36 32 5A 18 C3 74 0B 3A   += 03EC
0A90  59 18 B7 C2 D4 0A 21 4C 12 CD 75 0B CD 09 01 FE   += 0669
0AA0  33 D2 9C 0A FE 31 DA 9C 0A FE 31 C2 C1 0A 21 80   += 07B7
0AB0  00 22 56 18 3E 1A 32 5D 18 3E 1B 32 5A 18 C3 D1   += 0420
0AC0  0A 21 00 01 22 56 18 3E 0F 32 5D 18 3E 1B 32 5A   += 0295
0AD0  18 C3 74 0B 21 A4 12 CD 75 0B CD 09 01 FE 37 D2   += 065C
0AE0  DA 0A FE 31 DA DA 0A FE 31 C2 FF 0A 21 00 01 22   += 070F
0AF0  56 18 3E 1A 32 5D 18 3E 36 32 5A 18 C3 74 0B FE   += 04C5
0B00  32 C2 17 0B 21 00 02 22 56 18 3E 0E 32 5D 18 3E   += 02FA
0B10  36 32 5A 18 C3 74 0B FE 33 C2 2F 0B 21 00 02 22   += 048E
0B20  56 18 3E 0F 32 5D 18 3E 36 32 5A 18 C3 74 0B FE   += 04BA
0B30  34 C2 47 0B 21 00 02 22 56 18 3E 10 32 5D 18 3E   += 032E
0B40  28 32 5A 18 C3 74 0B FE 35 C2 5F 0B 21 00 04 22   += 04B4
0B50  56 18 3E 08 32 5D 18 3E 36 32 5A 18 C3 74 0B FE   += 04B3
0B60  36 C2 74 0B 21 00 04 22 56 18 3E 09 32 5D 18 3E   += 0358
0B70  28 32 5A 18 C9 7E B7 CA 83 0B 4F CD 06 01 23 7E   += 05E6
0B80  C3 76 0B C9 0E 1A CD 06 01 3A 54 18 B7 C2 96 0B   += 05C9
0B90  21 45 0D C3 99 0B 21 55 0D CD 75 0B 3A 53 18 B7   += 0506
0BA0  C2 A9 0B 21 65 0D C3 B7 0B FE 01 C2 B4 0B 21 6C   += 069B
0BB0  0D C3 B7 0B 21 79 0D CD 75 0B 3A 59 18 B7 C2 C7   += 0671
0BC0  0B 21 86 0D C3 CA 0B 21 8B 0D 3A 55 18 FE 01 C2   += 0578
0BD0  D5 0B 21 91 0D CD 75 0B CD 1F 0D CD 1F 0D 21 99   += 0598
0BE0  0D CD 75 0B 3A 5B 18 6F 26 00 CD AE 02 CD 1F 0D   += 0512
0BF0  21 AD 0D CD 75 0B 3A 5D 18 6F 26 00 CD AE 02 CD   += 05B6
0C00  1F 0D 21 C1 0D CD 75 0B 2A 56 18 CD AE 02 CD 1F   += 0569
0C10  0D 21 D5 0D CD 75 0B 3A 5E 18 E6 0F FE 01 C2 26   += 05E9
0C20  0C 0E 41 C3 46 0C FE 02 C2 30 0C 0E 42 C3 46 0C   += 04D3
0C30  FE 04 C2 3A 0C 0E 43 C3 46 0C FE 08 C2 44 0C 0E   += 0596
0C40  44 C3 46 0C 0E 3F CD 06 01 CD 1F 0D CD 1F 0D 3A   += 04A6
0C50  5C 18 FE 01 C2 6D 0C 3A 60 1B B7 C2 64 0C 21 39   += 05A3
0C60  0E C3 67 0C 21 58 0E CD 75 0B C3 91 0C 3A 5E 18   += 0528
0C70  E6 80 C2 7E 0C 21 DF 0D CD 75 0B C3 91 0C 3A 60   += 0706
0C80  18 B7 C2 8B 0C 21 FB 0D C3 8E 0C 21 16 0E CD 75   += 0635
0C90  0B CD 1F 0D CD 1F 0D 21 26 0D CD 75 0B 3A 5D 18   += 044D
0CA0  6F 26 00 3A 5B 18 5F 16 00 CD 02 0D 3A 5C 18 FE   += 043F
0CB0  01 C2 BA 0C 11 02 00 CD 02 0D ED 5B 56 18 7A B7   += 055F
0CC0  C2 D0 0C 06 03 CB 2C CB 1D 05 C2 C5 0C C3 F5 0C   += 06E2
0CD0  7A FE 01 C2 E1 0C CB 2C CB 1D CB 2C CB 1D C3 F5   += 089E
0CE0  0C 7A FE 02 C2 EE 0C CB 2C CB 1D C3 F5 0C 7A FE   += 085D
0CF0  04 C2 F5 0C 00 CD AE 02 21 3D 0D CD 75 0B CD 1F   += 05E8
0D00  0D C9 3E 10 D5 C1 11 00 00 EB F5 CB 43 CA 11 0D   += 06A1
0D10  09 CB 2C CB 1D CB 1A CB 1B F1 3D 20 ED EB C9 21   += 07C3
0D20  D3 16 CD 75 0B C9 53 70 65 69 63 68 65 72 6B 61   += 06FE
0D30  70 61 7A 69 74 61 65 74 20 3A 20 20 00 20 4B 42   += 04A9
rom  abs                                          checksum
0D40  79 74 65 73 00 4D 69 6E 69 2D 4C 61 75 66 77 65   += 05E3
0D50  72 6B 20 20 00 4D 61 78 69 2D 4C 61 75 66 77 65   += 053D
0D60  72 6B 20 20 00 46 4C 4F 32 20 20 00 46 4C 4F 31   += 0382
0D70  20 28 34 30 68 29 20 20 00 46 4C 4F 31 20 28 33   += 030A
0D80  30 68 29 20 20 00 46 4D 20 20 00 4D 46 4D 20 20   += 02F4
0D90  00 45 43 4D 41 37 30 20 00 41 6E 7A 61 68 6C 20   += 041B
0DA0  53 70 75 72 65 6E 20 20 20 20 3D 20 00 41 6E 7A   += 0483
0DB0  61 68 6C 20 53 65 6B 74 6F 72 65 6E 20 20 3D 20   += 053D
0DC0  00 42 79 74 65 73 2F 70 72 6F 20 53 65 6B 74 6F   += 05AD
0DD0  72 20 3D 20 00 4C 61 75 66 77 65 72 6B 20 00 56   += 04A6
0DE0  6F 72 64 65 72 73 65 69 74 65 20 77 69 72 64 20   += 062C
0DF0  66 6F 72 6D 61 74 69 65 72 74 00 52 75 65 63 6B   += 0637
0E00  73 65 69 74 65 20 77 69 72 64 20 66 6F 72 6D 61   += 0625
0E10  74 69 65 72 74 00 52 75 65 63 6B 73 65 69 74 65   += 063C
0E20  20 77 69 72 64 20 6D 69 74 20 53 53 4F 20 66 6F   += 054A
0E30  72 6D 61 74 69 65 72 74 00 62 65 69 64 65 20 53   += 05D4
0E40  65 69 74 65 6E 20 77 65 72 64 65 6E 20 66 6F 72   += 0621
0E50  6D 61 74 69 65 72 74 00 62 65 69 64 65 20 53 65   += 05C7
0E60  69 74 65 6E 20 77 65 72 64 65 6E 20 66 6F 72 6D   += 0629
0E70  61 74 69 65 72 74 0D 0A 53 65 69 74 65 20 31 20   += 050B
0E80  6D 69 74 20 53 53 4F 2C 20 53 65 69 74 65 20 30   += 04F5
0E90  20 6E 6F 72 6D 61 6C 00 0D 0A 41 43 48 54 55 4E   += 0483
0EA0  47 20 44 69 73 6B 65 74 74 65 20 77 69 72 64 20   += 059A
0EB0  66 6F 72 6D 61 74 69 65 72 74 0D 0A 53 74 61 72   += 05EE
0EC0  74 20 3D 20 22 4A 22 0D 0A 00 1A 0D 0A 20 55 6E   += 02AA
0ED0  69 76 65 72 73 61 6C 20 46 6F 72 6D 61 74 69 65   += 064D
0EE0  72 65 72 20 20 56 20 31 2E 32 20 2C 20 52 6F 6C   += 0429
0EF0  66 2D 44 69 65 74 65 72 20 4B 6C 65 69 6E 20 0D   += 0530
0F00  0A 20 28 43 29 20 31 39 38 34 2C 20 4D 75 65 6E   += 0395
0F10  63 68 65 6E 20 0D 0A 0D 0A 20 2D 2D 20 20 20 42   += 0308
0F20  69 74 74 65 20 53 79 73 74 65 6D 64 69 73 6B 65   += 066B
0F30  74 74 65 20 68 65 72 61 75 73 6E 65 68 6D 65 6E   += 0670
0F40  20 20 20 20 2D 2D 20 0D 0A 20 2D 2D 20 75 6E 64   += 02F2
0F50  20 7A 75 20 66 6F 72 6D 61 74 69 65 72 65 6E 64   += 062F
0F60  65 20 44 69 73 6B 65 74 74 65 20 65 69 6E 6C 65   += 05EF
0F70  67 65 6E 20 2D 2D 20 0D 0A 0D 0A 4D 69 6E 69 6C   += 03FB
0F80  61 75 66 77 65 72 6B 65 20 28 35 20 31 2F 34 22   += 04AD
0F90  29 20 3D 20 31 0D 0A 4D 61 78 69 6C 61 75 66 77   += 049C
0FA0  65 72 6B 65 20 28 38 22 20 20 20 20 29 20 3D 20   += 036F
0FB0  32 0D 0A 53 74 64 20 20 38 22 20 53 44 20 53 44   += 037C
0FC0  20 37 37 20 53 70 75 72 20 20 3D 20 33 0D 0A 45   += 0384
0FD0  43 4D 41 20 37 30 20 44 44 20 53 44 20 34 30 20   += 035B
0FE0  53 70 75 72 20 20 3D 20 34 0D 0A 4E 44 52 2D 20   += 03C3
0FF0  44 44 20 44 53 20 38 30 20 53 70 75 72 20 20 20   += 03F1
1000  20 20 3D 20 35 0D 0A 45 4E 44 45 20 20 20 20 20   += 02A5
1010  20 20 20 20 20 20 20 20 20 20 20 20 20 20 3D 20   += 021D
1020  36 0D 0A 0D 0A 00 1A 0D 0A 65 69 6E 66 61 63 68   += 0363
1030  65 20 53 63 68 72 65 69 62 64 69 63 68 74 65 20   += 05D6
1040  3D 20 31 0D 0A 64 6F 70 70 65 6C 74 65 20 53 63   += 04D8
rom  abs                                          checksum
1050  68 72 65 69 62 64 69 63 68 74 65 20 3D 20 32 0D   += 0537
1060  0A 0D 0A 00 1A 0D 0A 33 35 20 53 70 75 72 65 6E   += 0357
1070  20 20 3D 20 31 0D 0A 34 30 20 53 70 75 72 65 6E   += 03E6
1080  20 20 3D 20 32 0D 0A 37 30 20 53 70 75 72 65 6E   += 03EA
1090  20 20 3D 20 33 0D 0A 37 37 20 53 70 75 72 65 6E   += 03F2
10A0  20 20 3D 20 34 0D 0A 38 30 20 53 70 75 72 65 6E   += 03ED
10B0  20 20 3D 20 35 0D 0A 0D 0A 00 1A 0D 0A 45 69 6E   += 024D
10C0  73 65 69 74 69 67 65 73 20 4C 61 75 66 77 65 72   += 0653
10D0  6B 20 20 20 20 20 3D 20 31 0D 0A 44 6F 70 70 65   += 03A8
10E0  6C 74 73 65 69 74 69 67 65 73 20 4C 61 75 66 77   += 065C
10F0  65 72 6B 20 3D 20 32 0D 0A 44 6F 70 70 65 6C 74   += 04E0
1100  73 65 69 74 69 67 20 75 6E 64 20 53 53 4F 20 20   += 0541
1110  20 20 3D 20 33 0D 0A 0D 0A 00 1A 0D 0A 31 32 38   += 01CA
1120  20 42 79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F   += 05BE
1130  72 20 20 28 31 36 20 70 72 6F 20 54 72 61 63 6B   += 04C7
1140  29 20 3D 20 31 0D 0A 31 32 38 20 42 79 74 65 73   += 03B0
1150  20 70 72 6F 20 53 65 6B 74 6F 72 20 20 28 31 38   += 04DA
1160  20 70 72 6F 20 54 72 61 63 6B 29 20 3D 20 32 0D   += 046B
1170  0A 32 35 36 20 42 79 74 65 73 20 70 72 6F 20 53   += 04B2
1180  65 6B 74 6F 72 20 20 28 31 30 20 70 72 6F 20 54   += 04D3
1190  72 61 63 6B 29 20 3D 20 33 0D 0A 0D 0A 00 1A 0D   += 02CF
11A0  0A 32 35 36 20 42 79 74 65 73 20 70 72 6F 20 53   += 04B2
11B0  65 6B 74 6F 72 20 20 28 31 36 20 70 72 6F 20 54   += 04D9
11C0  72 61 63 6B 29 20 3D 20 31 0D 0A 35 31 32 20 42   += 0389
11D0  79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F 72 20   += 05EE
11E0  20 28 39 20 70 72 6F 20 54 72 61 63 6B 29 20 20   += 0470
11F0  3D 20 32 0D 0A 35 31 32 20 42 79 74 65 73 20 70   += 03F5
1200  72 6F 20 53 65 6B 74 6F 72 20 20 28 31 30 20 70   += 04D2
1210  72 6F 20 54 72 61 63 6B 29 20 3D 20 33 0D 0A 31   += 0417
1220  30 32 34 20 42 79 74 65 73 20 70 72 6F 20 53 65   += 0506
1230  6B 74 6F 72 20 28 35 20 70 72 6F 20 54 72 61 63   += 0558
1240  6B 29 20 20 3D 20 34 0D 0A 0D 0A 00 1A 0D 0A 31   += 01F5
1250  32 38 20 42 79 74 65 73 20 70 72 6F 20 53 65 6B   += 0545
1260  74 6F 72 20 28 32 36 20 70 72 6F 20 54 72 61 63   += 0520
1270  6B 29 20 3D 20 31 0D 0A 32 35 36 20 42 79 74 65   += 03AA
1280  73 20 70 72 6F 20 53 65 6B 74 6F 72 20 28 31 35   += 052A
1290  20 70 72 6F 20 54 72 61 63 6B 29 20 3D 20 32 0D   += 046B
12A0  0A 0D 0A 00 1A 0D 0A 32 35 36 20 42 79 74 65 73   += 0316
12B0  20 70 72 6F 20 53 65 6B 74 6F 72 20 28 32 36 20   += 04D9
12C0  70 72 6F 20 54 72 61 63 6B 29 20 3D 20 31 0D 0A   += 0454
12D0  35 31 32 20 42 79 74 65 73 20 70 72 6F 20 53 65   += 0508
12E0  6B 74 6F 72 20 28 31 34 20 70 72 6F 20 54 72 61   += 0525
12F0  63 6B 29 20 3D 20 32 0D 0A 35 31 32 20 42 79 74   += 03A4
1300  65 73 20 70 72 6F 20 53 65 6B 74 6F 72 20 28 31   += 055A
1310  35 20 70 72 6F 20 54 72 61 63 6B 29 20 3D 20 33   += 0494
1320  0D 0A 35 31 32 20 42 79 74 65 73 20 70 72 6F 20   += 0467
1330  53 65 6B 74 6F 72 20 28 31 36 20 70 72 6F 20 54   += 050C
1340  72 61 63 6B 29 20 3D 20 34 0D 0A 31 30 32 34 20   += 0379
1350  42 79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F 72   += 0610
rom  abs                                          checksum
1360  20 28 38 20 70 72 6F 20 54 72 61 63 6B 29 20 3D   += 048C
1370  20 35 0D 0A 31 30 32 34 20 42 79 74 65 73 20 70   += 03EA
1380  72 6F 20 53 65 6B 74 6F 72 20 28 39 20 70 72 6F   += 056B
1390  20 54 72 61 63 6B 29 20 3D 20 36 0D 0A 0D 0A 00   += 031F
13A0  1A 0D 0A 4C 61 75 66 77 65 72 6B 20 20 20 20 20   += 0412
13B0  20 20 20 20 20 20 20 20 20 41 20 3D 20 31 0D 0A   += 0226
13C0  4C 61 75 66 77 65 72 6B 20 20 20 20 20 20 20 20   += 0441
13D0  20 20 20 20 20 20 42 20 3D 20 32 0D 0A 4C 61 75   += 02EA
13E0  66 77 65 72 6B 20 20 20 20 20 20 20 20 20 20 20   += 037F
13F0  20 20 20 43 20 3D 20 33 0D 0A 4C 61 75 66 77 65   += 03CE
1400  72 6B 20 20 20 20 20 20 20 20 20 20 20 20 20 20   += 029D
1410  44 20 3D 20 34 0D 0A 4C 61 75 66 77 65 72 6B 20   += 046D
1420  41 20 52 75 65 63 6B 73 65 69 74 65 20 41 20 3D   += 0533
1430  20 35 0D 0A 4C 61 75 66 77 65 72 6B 20 42 20 52   += 04B1
```

```
1440  75 65 63 6B 73 65 69 74 65 20 42 20 3D 20 36 0D  += 04E4
1450  0A 4C 61 75 66 77 65 72 6B 20 43 20 52 75 65 63  += 055D
1460  6B 73 65 69 74 65 20 43 20 3D 20 37 0D 0A 4C 61  += 0460
1470  75 66 77 65 72 6B 20 44 20 52 75 65 63 6B 73 65  += 05EA
1480  69 74 65 20 44 20 3D 20 38 0D 0A 4E 45 55 53 54  += 0401
1490  41 52 54 20 20 20 20 20 3D 20 39 0D 0A 45 4E 44 45  += 0330
14A0  20 20 20 20 20 20 20 20 3D 20 30 0D 0A 0D 0A 00  += 01BB
14B0  1A 0D 0A 46 4C 4F 32 2D 4D 61 78 69 2F 4D 69 6E  += 0453
14C0  69 20 28 30 43 30 68 29 20 3D 20 31 0D 0A 46 4C  += 033C
14D0  4F 31 2D 4D 61 78 69 20 28 34 30 68 29 20 20 20  += 03D9
14E0  20 20 20 20 3D 20 32 0D 0A 46 4C 4F 31 2D 4D 69  += 031B
14F0  6E 69 20 28 33 30 68 29 20 20 20 20 20 20 3D 20  += 0330
1500  20 33 0D 0A 0D 0A 00 1A 0D 0A 44 69 73 6B 65 74  += 0316
1510  74 65 20 77 69 72 64 20 66 6F 72 6D 61 74 69 65  += 0626
1520  72 74 20 2E 2E 2E 20 62 69 74 74 65 20 77 61 72  += 0532
1530  74 65 6E 20 0D 0A 0D 0A 00 0D 0A 53 63 68 72 65  += 03A1
1540  69 62 73 63 68 75 74 7A 20 67 65 73 65 74 7A 74  += 0692
1550  0D 0A 4E 65 75 73 74 61 72 74 20 3D 20 54 61 73  += 0512
1560  74 65 20 57 20 64 72 75 65 63 6B 65 6E 0D 0A 00  += 04D8
1570  0D 0A 52 65 63 6F 72 64 20 54 79 70 20 69 73 74  += 0543
1580  20 66 61 6C 73 63 68 0D 0A 4E 65 75 73 74 61 72  += 058A
1590  74 20 3D 20 54 61 73 74 65 20 57 20 64 72 75 65  += 0539
15A0  63 6B 65 6E 0D 0A 00 0D 0A 52 65 63 6F 72 64 20  += 044E
15B0  6E 69 63 68 74 20 67 65 66 75 6E 64 65 0D 0A  += 0599
15C0  4E 65 75 73 74 61 72 74 20 3D 20 54 61 73 74 65  += 05D4
15D0  20 57 20 64 72 75 65 63 6B 65 6E 0D 0A 00 0D 0A  += 0416
15E0  43 52 43 20 46 65 68 6C 65 72 20 61 75 66 67 65  += 0576
15F0  74 72 65 74 65 6E 0D 0A 4E 65 75 73 74 61 72 74  += 05FF
1600  20 3D 20 54 61 73 74 65 20 57 20 64 72 75 65 63  += 0528
1610  6B 65 6E 0D 0A 00 0D 0A 43 50 55 20 7A 75 20 6C  += 03EF
1620  61 6E 67 73 61 6D 20 66 75 65 72 20 67 65 77 61  += 060D
1630  65 68 6C 74 65 73 20 46 6F 72 6D 61 74 2D 2D 20  += 0588
1640  44 61 74 65 6E 76 65 72 6C 75 73 74 0D 0A 68 6F  += 05EF
1650  65 68 65 72 65 72 20 54 61 6B 74 20 65 72 66 6F  += 05FB
1660  72 64 65 72 6C 69 63 68 0D 0A 4E 65 75 73 74 61  += 05D4
```

```
rom  abs                                                checksum
1670  72 74 20 3D 20 54 61 73 74 65 20 57 20 64 72 75  += 0546
1680  65 63 6B 65 6E 0D 0A 00 0D 0A 46 65 68 6C 65 72  += 048A
1690  20 61 75 66 67 65 74 72 65 74 65 6E 2E 20 4E 65  += 05BB
16A0  75 73 74 61 72 74 20 3D 20 54 61 73 74 65 20 57  += 0598
16B0  20 64 72 75 65 63 6B 65 6E 0D 0A 00 0D 0A 54 72  += 0465
16C0  61 63 6B 20 3D 20 00 2C 20 53 65 6B 74 6F 72 20  += 0490
16D0  3D 20 00 0D 0A 00 0D 0A 4C 61 75 66 77 65 72 6B  += 03CC
16E0  20 6E 69 63 68 74 20 52 45 41 44 59 20 2E 2E 2E  += 0475
16F0  20 62 69 74 74 65 20 70 72 75 66 65 6E 3A 0D  += 0594
1700  0A 20 31 2E 20 44 69 73 6B 65 74 74 65 20 66 61  += 04CD
1710  6C 73 63 68 68 65 72 75 6D 20 65 69 6E 67 65 6C  += 065F
1720  65 67 74 0D 0A 20 32 2E 20 4C 61 75 66 77 65 72  += 04CD
1730  6B 73 74 75 65 72 65 20 6E 69 63 68 74 20 67 65  += 0625
1740  73 63 68 6C 6F 73 73 65 6E 0D 0A 20 33 2E 20 66  += 04F0
1750  61 6C 73 63 68 65 20 4C 61 75 66 77 65 72 6B 73  += 0644
1760  6E 75 6D 6D 65 72 20 61 6E 67 65 67 65 62 65 6E  += 0650
1770  0D 0A 20 34 2E 20 4C 61 75 66 77 65 72 6B 73 6D  += 04DA
1780  6F 74 6F 72 20 64 72 65 68 74 20 73 69 63 68 20  += 05E2
1790  6E 69 63 68 74 0D 0A 20 35 2E 20 52 45 41 44 59  += 0445
17A0  2D 4C 65 69 74 75 6E 67 20 6E 69 63 68 74 20 6F  += 05CA
17B0  6B 0D 0A 0D 0A 4E 65 75 73 74 61 72 74 20 3D 20  += 046C
17C0  54 61 73 74 65 20 57 20 64 72 75 65 63 6B 65 6E  += 05E9
17D0  0D 0A 00 0D 0A 44 69 73 6B 65 74 74 65 20 77 69  += 046B
17E0  72 64 20 67 65 70 72 75 65 66 74 20 2E 2E 2E 20  += 0522
17F0  62 69 74 74 65 20 77 61 72 74 65 6E 0D 0A 0D 0A  += 04F7
1800  00 1A 0D 0A 53 79 73 74 65 6D 64 69 73 6B 65 74  += 053A
1810  74 65 20 69 6E 20 4C 61 75 66 77 65 72 6B 20 41  += 0592
1820  20 65 69 6E 6C 65 67 65 6E 2C 0D 0A 64 61 6E 6E  += 054B
1830  20 43 52 20 28 63 61 72 72 69 61 67 65 20 72 65  += 0532
1840  74 75 72 6E 29 20 64 72 75 65 63 6B 65 6E 2E 0D  += 059E
1850  0A 00 00 00 00 00 00 00 00 E5 00 00 00 00 00 00 00  += 00EF
1860  00 00 00 00 67 65 70 72 75 65 66 74 20 2E 2E 2E 20  += 042C
1870  62 69 74 74 65 20 77 61 72 74 65 6E 0D 0A 0D 0A  += 04F7
1880
```

```
Universal Formatierer V 1.2 , Rolf-Dieter Klein
(C) 1984, Muenchen

--  Bitte Systemdiskette herausnehmen   --
-- und zu formatierende Diskette einlegen --

Minilaufwerke (5 1/4")  = 1
Maxilaufwerke (8"    )  = 2
Std  8" SD SD 77 Spur   = 3
ECMA 70 DD SD 40 Spur   = 4
NDR- DD DS 80 Spur      = 5
ENDE                    = 6
```

Bild 8. Das Startmenü des Formatierers

```
FLO2-Maxi/Mini (0C0h) = 1
FLO1-Maxi (40h)       = 2
FLO1-Mini (30h)       = 3
```

Bild 9. FLO2 ist ein eigener Menüpunkt

```
Laufwerk               A = 1
Laufwerk               B = 2
Laufwerk               C = 3
Laufwerk               D = 4
Laufwerk A Rueckseite  A = 5
Laufwerk B Rueckseite  B = 6
Laufwerk C Rueckseite  C = 7
Laufwerk D Rueckseite  D = 8
NEUSTART    = 9
ENDE        = 0
```

Bild 10. Die Laufwerksauswahl

```
Mini-Laufwerk  FLO2  MFM

Anzahl Spuren    = 80
Anzahl Sektoren  = 5
Bytes/pro Sektor = 1024
Laufwerk A

beide Seiten werden formatiert
Seite 1 mit SSO, Seite 0 normal

Speicherkapazitaet :  800 KBytes

ACHTUNG Diskette wird formatiert
Start = "J"
```

Bild 11. Die Kontrollausgabe

```
Diskette wird formatiert ... bitte warten

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Diskette wird geprueft ... bitte warten

UUUUU
```

Bild 12. Der Formatiervorgang wird so angezeigt

```
35 Spuren  = 1
40 Spuren  = 2
70 Spuren  = 3
77 Spuren  = 4
80 Spuren  = 5
```

Bild 13. Menü zur Festlegung der Spurenzahl

```
Einseitiges Laufwerk      = 1
Doppeltseitiges Laufwerk  = 2
Doppeltseitig und SSO     = 3
```

Bild 14. Der Laufwerkstyp muß angegeben werden

```
einfache Schreibdichte = 1
doppelte Schreibdichte = 2
```

Bild 15. Die Schreibdichte wird hier festgelegt

```
128 Bytes pro Sektor  (16 pro Track) = 1
128 Bytes pro Sektor  (18 pro Track) = 2
256 Bytes pro Sektor  (10 pro Track) = 3
```

Bild 16. Bei einfacher Dichte, 5¼ Zoll, gibt es diese Auswahl

```
256 Bytes pro Sektor  (16 pro Track) = 1
512 Bytes pro Sektor  (9 pro Track)  = 2
512 Bytes pro Sektor  (10 pro Track) = 3
1024 Bytes pro Sektor (5 pro Track)  = 4
```

Bild 17. Bei doppelter Dichte, 5¼ Zoll, gibt es 4 Wahlmöglichkeiten

```
128 Bytes pro Sektor (26 pro Track) = 1
256 Bytes pro Sektor (15 pro Track) = 2
```

Bild 18. Zur einfachen Dichte, 8 Zoll

```
256 Bytes pro Sektor (26 pro Track) = 1
512 Bytes pro Sektor (14 pro Track) = 2
512 Bytes pro Sektor (15 pro Track) = 3
512 Bytes pro Sektor (16 pro Track) = 4
1024 Bytes pro Sektor (8 pro Track) = 5
1024 Bytes pro Sektor (9 pro Track) = 6
```

Bild 19. Zur doppelten Dichte, 8 Zoll

Rolf-Dieter Klein

68008: Arbeiten mit der Floppy Disk

Der Floppy-Controller Flo-2 ist schon ausführlich beschrieben. Er wird auch vom 68008-Vollausbau-System benutzt. In diesem Abschnitt geht es darum, wie er von der Software her angesprochen werden muß, damit man Sektoren auf einer Diskette beschreiben und lesen kann.

Ehe man auf einer Diskette lesen und schreiben kann, muß man diese formatiert haben. Dazu ist hier in *Bild 1* ein universelles Formatierprogramm als Hex Dump abgebildet. Es kann nicht als Quell Listing, also im Assembler-Klartext, abgebildet werden, weil das zuviel Platz in Anspruch nehmen würde.

So arbeitet UFORM68K

Bild 2 zeigt die Meldung, die entsteht, wenn UFORM68K mit der Bibliotheksfunktion angewählt wird. Die Adreß-Lage ist nur als Beispiel gedacht, das Programm ist beliebig im Speicher verschiebbar. Mit der Taste j wird UFORM angeworfen, es zeigt sich *Bild 3*. Je nach angeschlossener Floppy kann jetzt gewählt werden. Das Format NDR80 ist unser Standardformat, das auch von allen mc-Computern verstanden werden kann, soweit es die Spurlage und die Sektoreinteilung betrifft. Mit ihm kann man 800 KByte auf die Scheibe bekommen. Wenn Sie noch vor der Wahl eines Laufwerkes stehen, dann sollten Sie ein dazu passendes Laufwerk kaufen: nämlich 80 Spuren, zweiseitig, entweder in 5¼ Zoll oder einem der kleinen 3.XX-Formate. Von der Aufzeichnungstechnik her sind diese Typen alle einheitlich zu bedienen. Wählt man unser Standardformat, dann zeigt sich *Bild 4*. Allerdings werden die Rückseiten automatisch mitformatiert, wenn Sie dieses Menü über den Aufruf des NDR80-Formates erreicht haben. *Bild 5* zeigt sich nach Wahl des Laufwerkes und gibt Ihnen die Kontrolle, ob Sie wirklich dieses Format wollten, was Sie mit j bestätigen müssen.

Dabei darf die Diskette keinen Schreibschutz tragen. Es erscheinen sonst Fehlermeldungen.
Wenn man 8-Zoll-Disketten mit dem 68008 in doppelter Schreibdichte formatieren will, so erscheint die Fehlermeldung „CPU zu langsam". Das Gleiche gilt auch für andere Formate, wenn man den 68008 mit weniger als 8 MHz oder zuvielen WAIT-Zyklen betreibt. Der 68008 sollte beim Betrieb mit FLO2 mit genau drei Wait-Zyklen gefahren werden. Die Brücke muß dabei auf der dritten Position (von links nach rechts) stecken.

Die Routinen zum Lesen und Schreiben

Im Grundprogramm, Version 4.2, befinden sich bereits Unterprogramme für den Floppy-Betrieb.

Im Prinzip benötigt man davon genau zwei. Man will nämlich einen Sektor auf die Floppy schreiben können und man will ihn auch wieder lesen können. Ferner muß man noch angeben können, auf welcher Spur und auf welchem Laufwerk der Sektor steht, den man bearbeiten möchte. Dazu muß man noch einen Hinweis geben, in welcher Dichte der Sektor geschrieben werden soll und auf welcher Seite der Diskette sich der Sektor befindet. Dazu gibt es das Unterprogramm FLOPPY, das sowohl lesen als auch schreiben kann. Es muß die genannten Parameter beim Aufruf mitgeteilt bekommen.

Im Register A0 steht die Speicheradresse, die auf den Beginn der übertragenden oder zu überschreibenden Bytes zeigt. Also, woher die Daten kommen sollen oder wohin man sie schreiben will, das steht in A0.

Im Register D1.W steht der sogenannte Befehlscode. Wenn dort der Wert 0 steht, so soll das „Laufwerksparameter einstellen" bedeuten. Damit kann man die Steprate eines Laufwerks festlegen und ob beim Bearbeiten der Rückseite das Side-Select-Bit im Format berücksichtigt werden soll oder nicht (siehe FLO2-Beschreibung).

Diese Einstell-Informationen werden im Register D3.B übergeben. Die Steprate nimmt dabei die Werte 0 (schnellste Rate) bis 3 (langsamste Rate) an. Die langsamste Rate ist nach dem Einschalten zunächst eingestellt. Wenn man auf der Rückseite einer Diskette mit dem Side-Select-Bit arbeitet, so wie es beim NDR80-Format der Fall ist, so muß man den Wert $80 auf die Steprate addieren. Dann wird bei jedem nachfolgenden Zugriff auf die Rückseite der Floppy automatisch geprüft, ob die Stellung des Side-Select-Bits 1 ist.

Wenn im Register D1.W der Wert 1 steht, so soll ein Sektor gelesen werden. Der Wert 2 in D1.W sagt, daß ein Sektor geschrieben werden soll. Dazu werden weitere Parameter benötigt.

Im Register D2.B muß die Sektorennummer stehen. Beim NDR80-Format ist das ein Wert zwischen 1 und 5. Beim 8-Zoll-Format, einfache Dichte, ein Wert zwischen 1 und 26. Die Länge des Sektors wird vom Programm anhand der Formatierung auf der Diskette automatisch festgestellt. Beim NDR80-Format ist ein Sektor genau 1024 Byte lang, bei 8 Zoll, einfache Dichte, 128 Byte.

Im Register D3.B wird die Spurnummer übergeben. Bei 80-Spur-Laufwerken eine Zahl zwischen 0 und 79, bei 8-Zoll-Laufwerken zwischen 0 und 76. Hier fängt man also bei 0 an zu zählen, während man bei der Sektornummer normalerweise ab 1 zählt.

Dichte und Laufwerksnummer

Im Register D4.B muß ein kombinierter Laufwerksdichtecode stehen, der exakt der Bedeutung der Bits in Port $FFFFFFC4 des Floppy-Controllers entsprechen muß (siehe FLO2-Beschreibung).

Das Bit 7 bestimmt dabei, ob auf die Vorder- oder Rückseite zugegriffen wird.

```
-------------- Seite  1  -------- File uform68k ------------
rom  abs                                                 checksum
0000  55 AA 01 80 55 46 4F 52 4D 36 38 4B 00 00 09 9A     += 0465
0010  00 00 16 F6 01 00 00 00 00 00 00 00 00 00 00 00     += 010D
0020  00 00 00 00 48 E7 01 06 3E 3C 00 21 4E 41 4C DF     += 03BB
0030  60 80 4E 75 48 E7 7F FE 3E 3C 00 0C 4E 41 4C DF     += 068F
0040  7F FE 4E 75 48 E7 FF FE 3E 3C 00 14 4E 41 4C DF     += 07B4
0050  7F FF 4E 75 48 E7 7F FE 3E 3C 00 1C 4E 41 4C DF     += 073D
0060  7F FE 4E 75 48 E7 FF FE 3E 3C 00 19 4E 41 4C DF     += 07B9
0070  7F FF 4E 75 3E 3C 00 3B 4E 41 4E 75 48 E7 FF FE     += 0774
0080  3E 3C 00 2E 4E 41 4C DF 7F FF 4E 75 42 40 02 3C     += 0563
0090  00 FE 4E 75 30 3C FF FF 00 3C 00 01 4E 75 61 B4     += 0640
00A0  67 FC 61 B0 67 FC 4E 75 10 18 67 00 00 08 61 00     += 0592
00B0  FF 74 60 F4 4E 75 10 2E 00 0E 4A 2E 00 09 66 04     += 04C1
00C0  80 3C 00 10 4A 2E 00 07 66 04 80 3C 00 20 11 C0     += 0362
00D0  FF C4 4E 75 08 38 00 06 FF C4 67 F8 10 38 FF C0     += 07F5
00E0  4E 75 36 3C 00 64 10 38 FF C0 61 B2 61 C8 11 FC     += 06E9
00F0  00 0B FF C0 61 DE 08 00 00 07 67 92 51 CB FF E8     += 0714
0100  41 FA 0B B5 60 00 08 70 61 94 61 AA 11 FC 00 5B     += 063B
0110  FF C0 61 C0 4E 75 61 9E 11 EE 00 12 FF C2 41 EE     += 08A3
0120  00 3B 43 F8 FF C0 45 F8 FF C4 47 F8 FF C3 10 2E     += 0974
0130  00 0F C0 3C 00 80 E5 18 C0 3C 00 02 D0 3C 00 88     += 051A
0140  12 80 10 12 E3 18 6B 06 64 F8 10 D3 60 F4 10 11     += 05D4
0150  4E 75 61 00 FF 4A 61 00 FF 5E 41 EE 00 3B 43 F8     += 06D0
0160  FF C0 45 F8 FF C4 47 F8 FF C3 10 2E 00 0F C0 3C     += 0909
0170  00 80 E5 18 C0 3C 00 02 D0 3C 00 F4 12 80 10 12     += 052F
0180  E3 18 6B 06 64 F8 16 98 60 F4 10 11 C0 3C 00 44     += 062B
0190  4E 75 42 2E 00 06 41 FA 0B EA 61 00 FF 0C 10 38     += 051D
01A0  FF C0 61 00 FF 3E 42 2E 00 11 16 2E 00 0B 3F 03     += 046F
01B0  61 00 00 FC 61 00 01 D4 61 98 61 00 01 68 10 2E     += 0494
01C0  00 0C B0 3C 00 01 66 28 00 2E 00 80 00 0E 4A 2E     += 02BB
01D0  00 10 67 06 1D 7C 00 80 00 0F 61 00 01 AE 61 00     += 0316
01E0  FF 72 61 00 01 40 02 2E 00 7F 00 0E 42 2E 00 0F     += 034F
01F0  52 2E 00 11 10 2E 00 11 B0 2E 00 0B 67 04 61 00     += 0295
0200  FF 08 10 2E 00 06 B0 3C 00 28 66 08 42 2E 00 06     += 0343
0210  61 00 01 0A 52 2E 00 06 10 3C 00 46 61 00 FE 06     += 02E9
0220  36 1F 53 03 66 88 42 2E 00 06 41 FA 0D 5D 61 00     += 0415
0230  FE 78 61 00 FE AE 42 2E 00 11 16 2E 00 0B 3F 03     += 0495
0240  61 00 00 6C 61 00 02 AC 10 2E 00 0C B0 3C 00 01     += 0313
0250  66 20 00 2E 00 80 00 0E 4A 2E 00 10 67 06 1D 7C     += 02D0
0260  00 80 00 0F 61 00 02 8C 02 2E 00 7F 00 0E 42 2E     += 02AB
0270  00 0F 52 2E 00 11 10 2E 00 11 B0 2E 00 0B 67 04     += 0243
0280  61 00 FE 86 10 2E 00 06 B0 3C 00 28 66 08 42 2E     += 041B
0290  00 06 61 00 00 88 52 2E 00 06 10 3C 00 56 61 00     += 0278
02A0  FD 84 36 1F 53 03 66 96 61 00 FE 38 4E 75 4A 2E     += 05FA
02B0  00 08 67 00 FD DA 42 2E 00 07 10 2E 00 11 B0 3C     += 03F8
02C0  00 00 66 22 3D 7C 00 80 00 04 42 2E 00 09 42 2E     += 02AE
02D0  00 0C 1D 7C 00 10 00 0D 1D 7C 00 1B 00 0A 1D 7C     += 0219
02E0  00 28 00 0B 60 22 3D 7C 01 00 00 04 1D 7C 00 01     += 020D
02F0  00 09 42 2E 00 0C 1D 7C 00 10 00 0D 1D 7C 00 36     += 020A
0300  00 0A 1D 7C 00 28 00 0B 4E 75 41 EE 00 13 61 00     += 033C
0310  FD 6C 41 EE 00 13 61 00 FD 90 4E 75 41 FA 0C 68     += 070B
0320  60 00 FD 86 4A 00 67 00 FD 66 3F 00 41 FA 0C 40     += 05BD
0330  61 00 FD 76 42 40 10 2E 00 11 61 CE 41 FA 0C 3B     += 0556
0340  61 00 FD 66 42 40 10 2E 00 12 61 BE 61 CE 30 1F     += 0533
0350  41 FA 0C 05 08 00 00 02 67 04 41 FA 0B 7B 08 00     += 038A
0360  00 03 67 04 41 FA 0B 4D 08 00 00 04 67 04 41 FA     += 03B3
0370  0B 20 08 00 00 05 67 04 41 FA 0A F3 08 00 00 06     += 02E9
0380  67 04 41 FA 0A 2D 60 00 05 EE 41 EE 00 3B 4A 2E     += 0512
0390  00 09 66 0A 32 3C 00 0F 14 3C 00 FF 60 08 32 3C     += 031B
03A0  00 1F 14 3C 00 4E 10 C2 51 C9 FF FC 1D 7C 00 01     += 053E
03B0  00 12 12 2E 00 0D 3F 01 4A 2E 00 09 66 00 00 62     += 01E8
03C0  32 3C 00 05 42 18 51 C9 FF FC 10 FC 00 FE 10 EE     += 06EA
03D0  00 11 10 2E 00 0F C0 3C 00 80 66 06 10 FC 00 00     += 0352
03E0  60 04 10 FC 00 01 10 EE 00 12 30 2E 00 04 E0 58     += 041B
03F0  B0 3C 00 04 66 04 10 3C 00 03 10 C0 10 FC 00 F7     += 047C
0400  32 3C 00 0A 10 FC 00 FF 51 C9 FF FA 32 3C 00 05     += 0609
0410  10 FC 00 00 51 C9 FF FA 10 FC 00 FB 60 00 00 76     += 06FC
0420  32 3C 00 0B 42 18 51 C9 FF FC 32 3C 00 02 10 FC     += 0564
0430  00 F5 51 C9 FF FA 10 FC 00 FE 10 EE 00 11 10 2E     += 075F
0440  00 0F C0 3C 00 80 66 06 10 FC 00 00 60 04 10 FC     += 0473
0450  00 01 10 EE 00 12 30 2E 00 04 E0 58 B0 3C 00 04     += 039B
0460  66 04 10 3C 00 03 10 C0 10 FC 00 F7 32 3C 00 15     += 040F
0470  10 FC 00 4E 51 C9 FF FA 32 3C 00 0B 10 FC 00 00     += 05F2
0480  51 C9 FF FA 32 3C 00 02 10 FC 00 F5 51 C9 FF FA     += 0897
0490  10 FC 00 FB 32 2E 00 04 53 41 10 FC 00 E5 51 C9     += 060A
04A0  FF FA 10 FC 00 F7 4A 2E 00 09 66 06 10 3C 00 FF     += 0634
04B0  60 04 10 3C 00 4E 42 41 12 2E 00 0A 53 41 10 C0     += 032F
04C0  51 C9 FF FC 52 2E 00 12 32 1F 53 01 66 00 FE E8     += 0698
04D0  4A 2E 00 09 66 06 10 3C 00 FF 60 04 10 3C 00 4E     += 0336
04E0  32 3C 03 E7 10 C0 BF C8 65 00 04 74 51 C9 FF F6     += 079B
04F0  4E 75 61 00 FB AA 1D 7C 00 01 00 12 12 2E 00 0D     += 03C2
0500  3F 01 61 00 FC 12 61 00 FE 1C 52 2E 00 12 32 1F     += 040D
0510  53 01 66 EC 4E 75 41 FA 0C 17 61 00 FB 8C 61 00     += 0610
0520  FB 14 B0 3C 00 31 66 06 10 3C 00 23 60 32 B0 3C     += 0485
0530  00 32 66 06 10 3C 00 28 60 26 B0 3C 00 33 66 06     += 0323
0540  10 3C 00 46 60 1A B0 3C 00 34 66 06 10 3C 00 4D     += 0331
0550  60 0E B0 3C 00 35 66 06 10 3C 00 50 60 02 60 B6     += 040F
0560  1D 40 00 0B 4E 75 41 FA 0C 18 61 00 FB 3C 61 00     += 0483
0570  FA C4 B0 3C 00 31 66 06 42 2E 00 0C 60 24 B0 3C     += 0533
0580  00 32 66 08 1D 7C 00 01 00 0C 60 16 B0 3C 00 33     += 02DB
0590  66 0E 1D 7C 00 01 00 0C 1D 7C 00 01 00 10 60 02     += 0226
05A0  60 C4 4E 75 41 FA 0B 4B 61 00 FA FE 61 00 FA 86     += 07B2
05B0  B0 3C 00 31 66 06 10 3C 00 00 60 0E B0 3C 00 32     += 0361
05C0  66 06 10 3C 00 01 60 02 60 DA 1D 40 00 09 4E 75     += 037E
05D0  10 2E 00 07 B0 3C 00 00 66 00 00 E8 10 2E 00 09     += 02C6
05E0  B0 3C 00 00 66 00 00 62 41 FA 0B F6 61 00 FA BA     += 0605
05F0  61 00 FA 42 B0 3C 00 31 66 14 3D 7C 00 80 00 04     += 0471
0600  1D 7C 00 10 00 0D 1D 7C 00 1B 00 0A 60 36 B0 3C     += 02F6
0610  00 32 66 14 3D 7C 00 80 00 04 1D 7C 00 12 00 0D     += 02A1
0620  1D 7C 00 0A 00 0A 60 1C B0 3C 00 33 66 14 3D 7C     += 037B
0630  01 00 00 04 1D 7C 00 0A 00 0D 1D 7C 00 10 00 0A     += 0168
0640  60 02 60 AC 60 00 00 78 41 FA 0C 17 61 00 FA 5A     += 0559
0650  61 00 F9 E2 B0 3C 00 31 66 14 3D 7C 01 00 00 04     += 0491
0660  1D 7C 00 10 00 0D 1D 7C 00 36 00 0A 60 50 B0 3C     += 032B
0670  00 32 66 14 3D 7C 02 00 00 04 1D 7C 00 09 00 0D     += 021A
0680  1D 7C 00 36 00 0A 60 36 B0 3C 00 33 66 14 3D 7C     += 03C1
0690  02 00 00 04 1D 7C 00 0A 00 0D 1D 7C 00 28 00 0A     += 0181
06A0  60 1C B0 3C 00 34 66 14 3D 7C 04 00 00 04 1D 7C     += 0370
06B0  00 05 00 0D 1D 7C 00 36 00 0A 60 02 60 92 60 00     += 029F
06C0  01 0A 10 2E 00 09 B0 3C 00 00 66 00 00 48 41 FA     += 0327
06D0  0C 3B 61 00 F9 D4 61 00 F9 5C B0 3C 00 31 66 14     += 05C2
06E0  3D 7C 00 80 00 04 1D 7C 00 1A 00 0D 1D 7C 00 1B     += 02B1
06F0  00 0A 60 1C B0 3C 00 32 66 14 3D 7C 01 00 00 04     += 02DC
0700  1D 7C 00 0F 00 0D 1D 7C 00 1B 00 0A 60 02 60 C6     += 02FB
0710  60 00 00 B8 41 FA 0C 4D 61 00 F9 8E 61 00 F9 16     += 0604
0720  B0 3C 00 31 66 00 00 18 3D 7C 01 00 00 04 1D 7C     += 02F2
0730  00 1A 00 0D 1D 7C 00 36 00 0A 60 00 00 8E B0 3C     += 02DA
0740  00 32 66 00 00 18 3D 7C 02 00 00 04 1D 7C 00 0E     += 0216
0750  00 0D 1D 7C 00 36 00 0A 60 00 00 70 B0 3C 00 33     += 02D5
0760  66 00 00 16 3D 7C 02 00 00 04 1D 7C 00 0F 00 0D     += 01F0
0770  1D 7C 00 36 00 0A 60 52 B0 3C 00 34 66 14 3D 7C     += 03DE
```

Bild 1. Der Universal-Formatierer in Hex

```
0780  02 00 00 04 1D 7C 00 10 00 0D 1D 7C 00 28 00 0A   += 0187
0790  60 38 B0 3C 00 35 66 14 3D 7C 04 00 00 04 1D 7C   += 038D
07A0  00 08 00 0D 1D 7C 00 36 00 0A 60 1E B0 3C 00 36   += 028E
07B0  66 14 3D 7C 04 00 00 04 1D 7C 00 09 00 0D 1D 7C   += 0283
07C0  00 28 00 0A 60 04 60 00 FF 54 4E 75 10 3C 00 1A   += 0372
07D0  61 00 F8 52 4A 2E 00 07 66 06 41 FA 0D E8 60 04   += 052A
07E0  41 FA 0D F2 61 00 F8 C2 4A 2E 00 09 66 06 41 FA   += 067D
07F0  0D F4 60 04 41 FA 0D F3 10 2E 00 08 B0 3C 00 01   += 04D3
0800  66 04 41 FA 0D EB 61 00 F8 A0 61 00 FB 10 61 00   += 0663
0810  FB 0C 41 FA 0D E3 61 00 F8 90 42 40 10 2E 00 0B   += 05E6
0820  61 00 FA E8 61 00 FA F6 41 FA 0D E1 61 00 F8 7A   += 0890
0830  42 40 10 2E 00 0D 61 00 FA D2 61 00 FA E0 41 FA   += 0670
0840  0D DF 61 00 F8 64 30 2E 00 04 61 00 FA BE 61 00   += 0585
0850  FA CC 41 FA 0D DF 61 00 F8 50 10 2E 00 0E C0 3C   += 06DE
0860  00 0F B0 3C 00 01 66 06 10 3C 00 41 60 2A B0 3C   += 036B
0870  00 02 66 06 10 3C 00 42 60 1E B0 3C 00 04 66 06   += 02D6
0880  10 3C 00 43 60 12 B0 3C 00 08 66 08 10 3C 00 44   += 02F3
0890  60 00 00 06 10 3C 00 3F 61 00 F7 8A 61 00 FA 7E   += 04AC
08A0  61 00 FA 7A 10 2E 00 0C B0 3C 00 01 66 00 00 1A   += 038C
08B0  4A 2E 00 10 66 06 41 FA 0D DF 60 04 41 FA 0D F8   += 05BF
08C0  61 00 F7 E6 60 00 00 2A 10 2E 00 0E C0 3C 00 80   += 0490
08D0  66 0A 41 FA 0D 69 61 00 F7 D0 60 14 4A 2E 00 10   += 0545
08E0  66 06 41 FA 0D 75 60 04 41 EE 16 74 61 00 F7 BA   += 0658
08F0  61 00 FA 2A 61 00 FA 26 41 FA 0C AB 61 00 F7 AA   += 06FA
0900  42 40 42 41 10 2E 00 0D 12 2E 00 0B C0 C1 12 2E   += 035C
0910  00 0C B2 3C 00 01 66 02 D0 80 32 2E 00 04 B2 7C   += 0445
0920  00 80 66 04 E6 40 60 24 B2 7C 01 00 66 04 E4 40   += 0551
0930  60 1A B2 7C 02 00 66 04 E2 40 60 10 B2 7C 04 00   += 04D8
0940  66 06 4E 71 60 00 00 06 30 3C 00 00 61 00 F9 BC   += 0413
0950  41 FA 0C 6A 61 00 F7 52 61 00 F9 C2 4E 75 30 3C   += 06A6
0960  00 21 61 00 F7 00 61 00 F6 DC 41 FA 02 90 61 00   += 05DA
0970  F7 38 4E 71 60 FC 61 00 F7 30 41 FA 03 2D 61 00   += 069E
0980  F7 28 61 00 F6 B0 B0 3C 00 77 67 06 B0 3C 00 57   += 0639
0990  66 F0 2E 6E 00 00 60 00 00 2A 41 F9 00 00 80 00   += 0436
09A0  61 00 F6 D2 65 B8 20 09 08 80 00 00 00 D0 BC 00   += 0583
09B0  10 00 2C 40 D0 BC 00 00 1C 3B B0 88 64 A0 2D 4F   += 0517
09C0  00 00 30 3C 00 21 61 00 F6 9C 61 00 F6 78 41 FA   += 058A
09D0  06 01 61 00 F6 D4 61 00 F6 5C B0 3C 00 31 65 F6   += 065D
09E0  B0 3C 00 37 64 F0 42 2E 00 08 42 2E 00 0F 42 2E   += 03DE
09F0  00 10 B0 3C 00 31 66 00 00 1A 42 2E 00 07 61 00   += 0285
0A00  FB 16 61 00 FB 62 61 00 FB 9C 61 00 FB C4 60 00   += 0747
0A10  00 DA B0 3C 00 32 66 00 00 1C 1D 7C 00 01 00 07   += 031B
0A20  61 00 FA F4 61 00 FB 40 61 00 FB 7A 61 00 FB A2   += 07BF
0A30  60 00 00 B8 B0 3C 00 33 66 00 00 2C 1D 7C 00 01   += 0363
0A40  00 07 3D 7C 00 80 00 04 42 2E 00 09 1D 7C 00 1B   += 0271
0A50  00 0A 42 2E 00 0C 1D 7C 00 1A 00 0D 1D 7C 00 4D   += 022C
0A60  00 0B 60 00 00 86 B0 3C 00 34 66 00 00 38 42 2E   += 031F
0A70  00 07 1D 7C 00 01 00 08 1D 7C 00 01 00 10 3D 7C   += 020C
0A80  01 00 00 04 1D 7C 00 01 00 09 42 2E 00 0C 1D 7C   += 01BD
0A90  00 10 00 0D 1D 7C 00 36 00 0A 1D 7C 00 28 00 0B   += 01C2
0AA0  60 00 00 48 B0 3C 00 35 66 00 00 34 42 2E 00 07   += 02DA
0AB0  3D 7C 04 00 00 04 1D 7C 00 01 00 09 1D 7C 00 36   += 0233
0AC0  00 0A 1D 7C 00 05 00 0D 1D 7C 00 01 00 0C 1D 7C   += 01F4
0AD0  00 01 00 10 1D 7C 00 50 00 0B 60 00 00 0E B0 3C   += 025F
0AE0  00 36 66 00 00 06 60 00 01 08 41 FA 09 73 61 00   += 0323
0AF0  F5 B8 61 00 F5 40 B0 3C 00 31 66 00 00 0A 10 3C   += 051C
0B00  00 01 60 00 00 96 B0 3C 00 32 66 00 00 0A 10 3C   += 02D1
0B10  00 02 60 00 00 86 B0 3C 00 33 66 00 00 0A 10 3C   += 02C3
0B20  00 04 60 00 00 76 B0 3C 00 34 66 00 00 0A 10 3C   += 02B6
0B30  00 08 60 00 00 66 B0 3C 00 35 66 00 00 0A 10 3C   += 02AB
0B40  00 81 60 00 00 56 B0 3C 00 36 66 00 00 0A 10 3C   += 0315
0B50  00 82 60 00 00 46 B0 3C 00 37 66 00 00 0A 10 3C   += 0307
0B60  00 84 60 00 00 36 B0 3C 00 38 66 00 00 0A 10 3C   += 02FA

0B70  00 88 60 00 00 26 B0 3C 00 39 66 00 00 0A 60 00   += 0303
0B80  FE 42 60 00 00 16 B0 3C 00 30 66 00 00 0A 60 00   += 03A2
0B90  00 60 60 00 00 06 60 00 FF 52 1D 40 00 0E 4A 2E   += 035A
0BA0  00 10 67 0E 10 2E 00 0E C0 3C 00 80 67 04 1D 40   += 0315
0BB0  00 0F 61 00 FC 18 41 FA 09 BB 61 00 F4 EC 61 00   += 0625
0BC0  F4 74 B0 3C 00 4A 67 20 B0 3C 00 6A 67 1A B0 3C   += 05E8
0BD0  00 4E 67 00 FF 16 B0 3C 00 6E 67 00 FF 0E B0 3C   += 0584
0BE0  00 03 67 00 00 0C 60 CA 61 00 F5 A8 60 00 FE FC   += 05F8
0BF0  41 FA 03 C3 61 00 F4 B2 42 16 4E 75 0D 0A 2A 2A   += 058E
0C00  2A 20 46 65 68 6C 65 72 2C 20 7A 75 77 65 6E 69   += 058E
0C10  67 20 52 41 4D 20 2A 2A 0D 0A 0D 0A 44 61 73   += 034B
0C20  20 50 72 6F 67 72 61 6D 6D 20 62 65 6E 6F 65 74   += 0602
0C30  69 67 74 20 6D 69 6E 64 65 73 74 65 6E 73 0D 0A   += 05B5
0C40  0D 0A 38 4B 20 65 69 67 65 6E 65 6E 20 52 41 4D   += 0495
0C50  2C 20 62 65 67 69 6E 6E 65 6E 64 20 62 65 69 6D   += 05B3
0C60  20 0D 0A 0D 0A 54 65 78 74 73 70 65 69 63 68 65   += 04D4
0C70  72 2C 20 64 65 72 20 75 65 62 65 72 73 63 68 72   += 05DC
0C80  69 65 62 65 6E 0D 0A 0D 0A 77 69 72 64 2E 0D 0A   += 042C
0C90  42 69 74 74 65 20 20 52 45 53 45 54 20 64 72 75   += 0526
0CA0  65 63 6B 65 6E 2E 0D 0A 00 0D 0A 77 65 69 74 65   += 0480
0CB0  72 20 3D 20 77 20 00 0D 0A 46 65 68 6C 65 72 3A   += 042D
0CC0  20 4C 61 75 66 77 65 72 6B 20 6D 65 6C 64 65 74   += 05FC
0CD0  20 6E 69 63 68 74 20 52 45 41 44 59 2E 0D 0A 42   += 0452
0CE0  69 74 74 65 20 70 72 75 65 66 65 6E 3A 0D 0A 20   += 053C
0CF0  31 2E 20 44 69 73 6B 65 74 74 65 20 66 61 6C 73   += 0582
0D00  63 68 20 65 69 6E 67 65 6C 65 67 74 2E 0D 0A 20   += 0504
0D10  32 2E 20 4C 61 75 66 77 65 72 6B 74 75 65 72 65   += 05E6
0D20  20 6F 66 66 65 6E 2E 0D 0A 20 33 2E 20 46 61 6C   += 0427
0D30  73 63 68 65 20 4C 61 75 66 77 65 72 6B 73 6E 75   += 065A
0D40  6D 6D 65 72 2E 0D 0A 20 34 2E 20 4D 6F 74 6F 72   += 04A9
0D50  20 64 72 65 68 74 20 73 69 63 68 20 6E 69 63 68   += 05C0
0D60  74 2E 0D 0A 20 35 2E 20 52 45 41 44 59 2D 4C 65   += 03AF
0D70  69 74 75 6E 67 20 6E 69 63 68 74 20 6F 6B 2E 0D   += 0592
0D80  0A 00 1A 0D 0A 44 69 73 6B 65 74 74 65 20 77 69   += 0478
0D90  72 64 20 66 6F 72 6D 61 74 69 65 72 74 2E 20 42   += 05C3
0DA0  69 74 74 65 20 77 61 72 74 65 6E 2E 0D 0A 0D 0A   += 04C3
0DB0  00 0D 0A 46 65 68 6C 65 72 3A 20 53 63 68 72 65   += 04BC
0DC0  69 62 73 63 68 75 74 7A 20 67 65 73 65 74 7A 74   += 0692
0DD0  0D 0A 44 65 72 20 46 65 68 6C 65 72 20 65 6E 74   += 050F
0DE0  73 74 65 68 74 20 61 75 63 68 2C 20 77 65 6E 6E   += 05ED
0DF0  0D 0A 64 69 65 20 44 69 73 6B 65 74 74 65 20 66   += 052C
0E00  61 6C 73 63 68 20 68 65 72 75 6D 20 0D 0A 65 69   += 0551
0E10  6E 67 65 6C 65 67 74 20 77 75 72 64 65 2C 20 6F   += 05E8
0E20  64 65 72 20 64 69 65 0D 0A 4C 61 75 66 77 65 72   += 057A
0E30  6B 73 74 75 65 72 65 20 6F 66 66 65 6E 20 69 73   += 062D
0E40  74 0D 0A 75 6E 64 20 64 61 73 20 4C 61 75 66 77   += 0549
0E50  65 72 6B 20 64 65 6E 6E 6F 63 68 0D 0A 52 45 41   += 0530
0E60  44 59 20 6D 65 6C 64 65 74 2E 0D 0A 00 0D 0A 46   += 03DA
0E70  65 68 6C 65 72 3A 20 52 65 63 6F 72 64 20 54 79   += 05B6
0E80  70 20 69 73 74 20 66 61 6C 73 63 68 2E 0D 0A 00   += 04B6
0E90  0D 0A 46 65 68 6C 65 72 3A 20 52 65 63 6F 72 64   += 0526
0EA0  20 6E 69 63 68 74 20 67 65 66 75 6E 64 65 6E 2E   += 05D0
0EB0  0D 0A 00 0D 0A 46 65 68 6C 65 72 3A 20 43 52 43   += 03B6
0EC0  2D 46 65 68 6C 65 72 20 61 75 66 67 65 74 72 65   += 05F6
0ED0  74 65 6E 2E 0D 0A 00 0D 0A 43 50 55 20 7A 75 20   += 03BA
0EE0  6C 61 6E 67 73 61 6D 20 66 75 65 72 20 67 65 77   += 0618
0EF0  61 65 68 6C 74 65 73 20 46 6F 72 6D 61 74 2C 0D   += 05A8
0F00  0A 44 61 74 65 6E 76 65 72 6C 75 73 74 2E 20 0D   += 0566
0F10  0A 48 6F 65 68 65 72 65 20 54 61 6B 74 66 72 65   += 05BB
0F20  71 75 65 6E 7A 20 6E 6F 65 74 69 67 2E 0D 0A 4F   += 056D
0F30  64 65 72 20 41 6E 7A 61 68 6C 20 64 65 72 20 57   += 058B
0F40  61 69 74 2D 5A 79 6B 6C 65 6E 20 7A 75 20 67 72   += 05F0
0F50  6F 73 73 2E 0D 0A 00 0D 0A 46 65 68 6C 65 72 20   += 0427
```

```
0F60  61 75 66 67 65 74 72 65 74 65 6E 0D 0A 00 0D 0A   += 04C8
0F70  20 53 70 75 72 20 3D 20 00 0D 0A 20 53 65 6B 74   += 0415
0F80  6F 72 20 3D 20 00 0D 0A 00 0D 0A 44 69 73 6B 65   += 037C
0F90  74 74 65 20 77 69 72 64 20 67 65 70 72 75 65 66   += 0631
0FA0  74 2E 20 42 69 74 74 65 20 77 61 72 74 65 6E 2E   += 0599
0FB0  0D 0A 0D 0A 00 0D 0A 46 6F 72 6D 61 74 70 72 6F   += 03FF
0FC0  67 72 61 6D 6D 20 62 65 65 6E 64 65 74 2E 0D 0A   += 0550
0FD0  00 1A 0D 0A 55 6E 69 76 65 72 73 61 6C 20 46 6F   += 04BF
0FE0  72 6D 61 74 69 65 72 65 72 20 56 20 31 2E 32 20   += 0512
0FF0  0D 0A 28 43 29 20 31 39 38 35 20 52 6F 6C 66 2D   += 0382
1000  44 69 65 74 65 72 20 4B 6C 65 69 6E 20 20 0D 0A   += 04C7
1010  0D 0A 2D 2D 2D 20 20 20 20 20 20 44 69 73 6B 65   += 034E
1020  74 74 65 20 20 20 20 20 2D 2D 2D 20 20 0D 0A 2D   += 02F8
1030  2D 2D 20 20 20 20 20 20 65 69 6E 6C 65 67 65 6E   += 0461
1040  20 20 20 20 20 2D 2D 2D 20 20 0D 0A 0D 0A 4D 69   += 024B
1050  6E 69 6C 61 75 66 77 65 72 6B 20 28 35 20 31 2F   += 0535
1060  34 22 29 20 3D 20 31 0D 0A 4D 61 78 69 6C 61 75   += 0415
1070  66 77 65 72 6B 20 28 38 22 20 20 20 20 29 20 3D   += 03C7
1080  20 32 0D 0A 53 74 64 20 38 22 20 20 53 44 20 53   += 0358
1090  53 20 37 37 20 53 70 75 72 20 3D 20 33 0D 0A 45   += 03B7
10A0  43 4D 41 20 37 30 20 44 44 20 53 53 20 34 30 20   += 036A
10B0  53 70 75 72 20 3D 20 34 0D 0A 4E 44 52 38 30 20   += 03DE
10C0  20 20 44 44 20 44 53 20 38 30 20 53 70 75 72 20   += 03F1
10D0  3D 20 35 0D 0A 45 4E 44 45 20 20 20 20 20 20 20   += 02A5
10E0  20 20 20 20 20 20 20 20 20 20 3D 20 36 0D 0A      += 020A
10F0  00 1A 0D 0A 65 69 6E 66 61 63 68 65 20 53 63 68   += 04A2
1100  72 65 69 62 64 69 63 68 74 65 20 3D 20 31 0D 0A   += 04D8
1110  64 6F 70 70 65 6C 74 65 20 53 63 68 72 65 69 62   += 063D
1120  64 69 63 68 74 65 20 3D 20 32 0D 0A 0D 0A 00 1A   += 0368
1130  0D 0A 33 35 20 53 70 75 72 65 6E 20 3D 20 31 0D   += 03D7
1140  0A 34 30 20 53 70 75 72 65 6E 20 3D 20 32 0D 0A   += 03D1
1150  37 30 20 53 70 75 72 65 6E 20 3D 20 33 0D 0A 37   += 0402
1160  37 20 53 70 75 72 65 6E 20 3D 20 34 0D 0A 38 30   += 0404
1170  20 53 70 75 72 65 6E 20 3D 20 35 0D 0A 0D 0A 00   += 037D
1180  1A 0D 0A 45 69 6E 73 65 69 74 69 67 65 73 20 4C   += 0516
1190  61 75 66 77 65 72 6B 20 20 20 20 20 3D 20 31 0D   += 0430
11A0  0A 44 6F 70 70 65 6C 74 73 65 69 74 69 67 65 73   += 063F
11B0  20 4C 61 75 66 77 65 72 6B 20 3D 20 32 0D 0A 44   += 046B
11C0  6F 70 70 65 6C 74 73 65 69 74 69 67 20 75 6E 64   += 0680
11D0  20 53 53 4F 20 20 20 20 3D 20 33 0D 0A 0D 0A 00   += 0253
11E0  1A 0D 0A 31 32 38 20 42 79 74 65 73 20 70 72 6F   += 0464
11F0  20 53 65 6B 74 6F 72 20 28 31 36 20 70 72 6F 20   += 04D8
1200  54 72 61 63 6B 29 20 3D 20 31 0D 0A 31 32 38 20   += 039E
1210  42 79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F 72   += 0610
1220  20 28 31 38 20 70 72 6F 20 54 72 61 63 6B 29 20   += 0480
1230  3D 20 32 0D 0A 32 35 36 20 42 79 74 65 73 20 70   += 03FA
1240  72 6F 20 53 65 6B 74 6F 72 20 28 31 30 20 70 72   += 0524
1250  6F 20 54 72 61 63 6B 29 20 3D 20 33 0D 0A 0D 0A   += 038B
1260  00 1A 0D 0A 32 35 36 20 42 79 74 65 73 20 70 72   += 03F7
1270  6F 20 53 65 6B 74 6F 72 20 28 31 36 20 70 72 6F   += 0527
1280  20 54 72 61 63 6B 29 20 3D 20 31 0D 0A 35 31 32   += 039B
1290  20 42 79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F   += 05BE
12A0  72 20 28 20 39 20 70 72 6F 20 54 72 61 63 6B 29   += 04C2
12B0  20 3D 20 32 0D 0A 35 31 32 20 42 79 74 65 73 20   += 03A5
12C0  70 72 6F 20 53 65 6B 74 6F 72 20 28 31 30 20 70   += 0522
12D0  72 6F 20 54 72 61 63 6B 29 20 3D 20 33 0D 0A 31   += 0417
12E0  30 32 34 20 42 79 74 65 73 20 70 72 6F 20 53 65   += 0506
12F0  6B 74 6F 72 28 20 35 20 70 72 6F 20 54 72 61 63   += 0558
1300  6B 29 20 3D 20 34 0D 0A 0D 0A 00 1A 0D 0A 31 32   += 0207
1310  38 20 42 79 74 65 73 20 70 72 6F 20 53 65 6B 74   += 0587
1320  6F 72 20 28 32 36 20 70 72 6F 20 54 72 61 63 6B   += 0517
1330  29 20 3D 20 31 0D 0A 32 35 36 20 42 79 74 65 73   += 03B2
1340  20 70 72 6F 20 53 65 6B 74 6F 72 20 28 31 35 20   += 04D7
1350  70 72 6F 20 54 72 61 63 6B 29 20 3D 20 32 0D 0A   += 0455
1360  0D 0A 00 1A 0D 0A 32 35 36 20 42 79 74 65 73 20   += 032C
1370  70 72 6F 20 53 65 6B 74 6F 72 20 28 32 36 20 70   += 0529
1380  72 6F 20 54 72 61 63 6B 29 20 3D 20 31 0D 0A 35   += 0419
1390  31 32 20 42 79 74 65 73 20 70 72 6F 20 53 65 6B   += 053E
13A0  74 6F 72 20 28 31 34 20 70 72 6F 20 54 72 61 63   += 051D
13B0  6B 29 20 3D 20 32 0D 0A 35 31 32 20 42 79 74 65   += 03A6
13C0  73 20 70 72 6F 20 53 65 6B 74 6F 72 20 28 31 35   += 052A
13D0  20 70 72 6F 20 54 72 61 63 6B 29 20 3D 20 33 0D   += 046C
13E0  0A 35 31 32 20 42 79 74 65 73 20 70 72 6F 20 53   += 04AD
13F0  65 6B 74 6F 72 20 28 31 36 20 70 72 6F 20 54 72   += 052B
1400  61 63 6B 29 20 3D 20 34 0D 0A 31 30 32 34 20 42   += 0349
1410  79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F 72 28   += 05F6
1420  20 38 20 70 72 6F 20 54 72 61 63 6B 29 20 3D 20   += 0484
1430  35 0D 0A 31 30 32 34 20 42 79 74 65 73 20 70 72   += 043C
1440  6F 20 53 65 6B 74 6F 72 28 20 39 20 70 72 6F 20   += 0519
1450  54 72 61 63 6B 29 20 3D 20 36 0D 0A 0D 0A 00 1A   += 0319
1460  0D 0A 4C 61 75 66 77 65 72 6B 20 20 20 20 20 20   += 0418
1470  20 20 20 20 20 20 41 20 3D 20 31 0D 0A 4C 61 75   += 02E8
1480  66 77 65 72 6B 20 20 20 20 20 20 20 20 20 20 20   += 037F
1490  20 42 20 3D 20 32 0D 0A 4C 61 75 66 77 65 72 6B   += 0469
14A0  20 20 20 20 20 20 20 20 20 20 20 20 43 20 3D 20   += 0240
14B0  33 0D 0A 4C 61 75 66 77 65 72 6B 20 20 20 20 20   += 042B
14C0  20 20 20 20 20 20 20 44 20 3D 20 34 0D 0A 4C 61   += 0299
14D0  75 66 77 65 72 6B 20 52 75 65 63 6B 73 65 69 74   += 0663
14E0  65 20 41 20 3D 20 35 0D 0A 4C 61 75 66 77 65 72   += 0465
14F0  6B 20 52 75 65 63 6B 73 65 69 74 65 20 42 20 3D   += 055E
1500  20 36 0D 0A 4C 61 75 66 77 65 72 6B 20 52 75 65   += 04FA
1510  63 6B 73 65 69 74 65 20 43 20 3D 20 37 0D 0A 4C   += 0462
1520  61 75 66 77 65 72 6B 20 52 75 65 63 6B 73 65 69   += 0650
1530  74 65 20 44 20 3D 20 38 0D 0A 4E 45 55 53 54 41   += 03D9
1540  52 54 20 20 20 20 20 20 20 20 20 20 20 20 20 20   += 0266
1550  3D 20 39 0D 0A 45 4E 44 45 20 20 20 20 20 20 20   += 02A9
1560  20 20 20 20 20 20 20 20 20 20 20 20 3D 20 30 0D   += 0204
1570  0A 00 0D 0A 41 43 48 54 55 4E 47 2E 20 44 69      += 0333
1580  73 6B 65 74 74 65 20 77 69 72 64 20 66 6F 72 6D   += 063A
1590  61 74 69 65 72 74 2E 0D 0A 53 74 61 72 74 20 3D   += 0539
15A0  20 4A 0D 0A 00 53 70 65 69 63 68 65 72 6B 61 70   += 04F0
15B0  61 7A 69 74 61 65 74 20 3A 20 20 00 20 4B 42 79   += 04B2
15C0  74 65 73 00 4D 69 6E 69 2D 4C 61 75 66 77 65 72   += 05DC
15D0  6B 20 20 00 4D 61 78 69 2D 4C 61 75 66 77 65 72   += 053D
15E0  6B 20 20 00 46 4D 20 20 00 4D 46 4D 20 20 00 45   += 02E3
15F0  43 4D 41 37 30 20 00 41 6E 7A 61 68 6C 20 53 70   += 0499
1600  75 72 65 6E 20 20 20 3D 20 00 41 6E 7A 61 68      += 0489
1610  6C 20 53 65 6B 74 6F 72 65 6E 20 20 3D 20 00 42   += 04B6
1620  79 74 65 73 2F 70 72 6F 20 53 65 6B 74 6F 72 20   += 05FD
1630  3D 20 00 4C 61 75 66 77 65 72 6B 20 00 56 6F 72   += 04F5
1640  64 65 72 73 65 69 74 65 20 77 69 72 64 20 66 6F   += 0620
1650  72 6D 61 74 69 65 72 74 00 52 75 65 63 6B 73 65   += 063A
1660  69 74 65 20 77 69 72 64 20 66 6F 72 6D 61 74 69   += 062A
1670  65 72 74 00 52 75 65 63 6B 73 65 69 74 65 20 77   += 05F6
1680  69 72 64 20 6D 69 74 20 53 53 4F 20 66 6F 72 6D   += 0592
1690  61 74 69 65 72 74 00 62 65 69 64 65 20 53 65 69   += 05C3
16A0  74 65 6E 20 77 65 72 64 65 6E 20 66 6F 72 6D 61   += 0621
16B0  74 69 65 72 74 00 62 65 69 64 65 20 53 65 69 74   += 05D6
16C0  65 6E 20 77 65 72 64 65 6E 20 66 6F 72 6D 61 74   += 0621
16D0  69 65 72 74 0D 0A 53 65 69 74 65 20 31 20 6D 69   += 050C
16E0  74 20 53 53 4F 2C 20 53 65 69 74 65 20 30 20 6E   += 04AD
16F0  6F 72 6D 61 6C 00 FF FF FF FF FF FF FF FF FF FF   += 0C11
1700
```

```
Bibliothek

Name   : UFORM68K
Start  : 00E99A
Laenge : 0016F6

Starten J = JA
cr = weiter, M = Menue
```

Bild 2. Der Bibliotheksinhalt zeigt UFORM

Ist es 0, so wird die Vorderseite verwendet. Mit Bit 6 kann man den Laufwerksmotor schalten. Dies sollte aber nur durch direktes Schreiben auf Port $FFFFFFC4 tun, nicht bei einem Floppy-Zugriff nebenbei. Das Bit 6 wird also normalerweise auf 0 (Motor an) stehen. Bit 5 gibt an, ob ein Mini-Laufwerk oder ein Maxi-Laufwerk angesteuert werden soll. Wenn es 1 ist, so wird das Mini-Format verwendet. Mit Bit 4 wird die Schreibdichte eingestellt. Ist es 0, so wird doppelte Dichte verwendet (MFM), ist es 1, so wird die einfache Schreibdichte verwendet. Damit ergeben sich 4 verschiedene Übertragungsraten. Minilaufwerke mit einfacher Dichte werden mit 125 KBit pro Sekunde bedient, Minilaufwerke mit doppelter Dichte und Maxilaufwerke mit einfacher Dichte werden mit 250 KBit pro Sekunde bedient. Maxilaufwerke mit doppelter Dichte müssen mit 500 KBit pro Sekunde bedient werden. Das ist der Grund, weshalb der 68008 bei diesem Format in der 8-MHz-Version Probleme bekommt, denn er hat dann nur noch 16 Mikrosekunden Zeit, um ein einzelnes Byte abzuholen und wegzustecken. Weil er dabei den Status abfragen muß, gelingt ihm das nicht. Erst mit der geplanten CPU 68 K/16 bei 12 MHz gibt es keine Probleme mehr.
In den Bits 0 bis 3, des Registers D4.B muß der Laufwerkscode stehen. Dabei ist der Wert 1 für das Laufwerk A, 2 für B, 4 für C und 8 für D zuständig.
Will man mehr als 4 Laufwerke ansprechen, so muß man binäre Codierung verwenden, die aber nur wenige Laufwerke auswerten können.

Nach Durchführung des Unterprogramms FLOPPY erscheint im Register D0.W eine codierte Mitteilung über den Verlauf der Aktion. Wenn das Register den Wert 0 hat, so war alles fehlerfrei.

Wenn der Wert $FFFF erscheint, so hat es Fehler gegeben, beim Lesen z. B. einen CRC-Fehler oder der eingestellte Laufwerkscode war falsch. Beim Schreiben war in solch einem Fall entweder der Dichtecode falsch, oder die CPU wird mit einer falschen Taktrate (kleiner als 8 MHz) betrieben (dann entsteht ein Datenverlust, den das Unterprogramm feststellen kann).

Bevor wir ein paar Beispiele behandeln, noch ein wichtiges Unterprogramm: GETFLOP. Damit ist es möglich, die Schreibdichte einer Diskette im Laufwerk automatisch festzustellen. Wenn man bei GETFLOP in Register D4 den Laufwerkscode angibt ohne die restlichen Bits 4 bis 7 zu setzen, so steht in diesen Bits anschließend der richtige Dichte-Code (wenn die Diskette auch formatiert war). Damit kann man automatische Starts von Programmen auf Diskette durchführen. So wird diese Routine vom Grundprogramm dazu verwendet, den Menüpunkt Floppy-Start zu realisieren. *Bild 6* zeigt den Einstieg. Es wird der Sektor 1 auf Spur 0 von Laufwerk A auf die Adresse $9800 geladen, egal in welchem Format dieser Sektor geschrieben ist. Dann wird auf Adresse $9800 gestartet, wenn das Grundprogramm dort als erstes den Befehl NOP findet.

```
Universal Formatierer V 1.2
(C) 1985 Rolf-Dieter Klein

---      Diskette      ---
---      einlegen      ---

Minilaufwerk (5 1/4")  = 1
Maxilaufwerk (8"    )  = 2
Std 8"  SD SS 77 Spur  = 3
ECMA 70 DD SS 40 Spur  = 4
NDR80   DD DS 80 Spur  = 5
ENDE                   = 6
```

Bild 3. Das Hauptmenü des Formatierers

```
Laufwerk              A = 1
Laufwerk              B = 2
Laufwerk              C = 3
Laufwerk              D = 4
Laufwerk Rueckseite A = 5
Laufwerk Rueckseite B = 6
Laufwerk Rueckseite C = 7
Laufwerk Rueckseite D = 8
NEUSTART              = 9
ENDE                  = 0
```

Bild 4. Zur Laufwerksauswahl dieses Menü

```
Mini-Laufwerk  MFM

Anzahl Spuren     = 80
Anzahl Sektoren   = 5
Bytes/pro Sektor = 1024
Laufwerk A

beide Seiten werden formatiert
Seite 1 mit SSO, Seite 0 normal

Speicherkapazitaet :  800 KBytes

ACHTUNG. Diskette wird formatiert.
Start = J
```

Bild 5. Das angewählte Format wird zur Kontrolle angezeigt

Ein Beispiel

Wir wollen nun ein Programm schreiben, das sich selbst auf Diskette auf genau diesen Sektor schreibt und gleichzeitig ein kleines Schildkrötenprogramm realisiert, das sich dann beim Aufruf über Floppy-Start gleich meldet. Zunächst formatieren Sie dazu bitte eine neue Diskette, bevor Sie mit dem Eintippen des Programms beginnen. Dabei spielt das verwendete Format keine besondere Rolle, es muß natürlich für Ihr Laufwerk passen.

Jetzt wählen Sie im Menü TEXTSTART den Start $9000 neu aus, denn das Formatierprogramm hat dort diverse Informationen hinterlassen, die nicht mehr benötigt werden. *Bild 7* zeigt das Programm, wie es im Editor erscheint. Tippen Sie das Programm ein.

Wenn Sie das Programm übersetzt haben, so wird es mit der Adresse „Speichern" gestartet. Dann muß die Floppy kurz klicken, das Programm ist abgespeichert. Dabei wird zunächst mit GETFLOP das Format bestimmt, das kann bis zu 1 s dauern (es geht schneller, wenn man in D4 gleich den Dichtecode mit einträgt). Danach wird das Programm ab Adresse „Start" auf die Floppy gebracht. Dabei wird es auf die Spur 0, Sektor 1 geschrieben. Wieviele Bytes geschrieben werden, hängt von der Formatierung ab.

Da das Programm in Bild 7 aber kleiner als 128 Bytes ist, wird es auf jeden Fall ganz abgespeichert. Dann wird noch das Textgebiet auf den Sektor 2 geschrieben, so daß der Klartext auch gerettet wird.

Dabei muß man aufpassen, denn der Text ist fast 1024 Bytes lang, er wird nur dann ganz abgespeichert, wenn man genügend große Sektoren verwendet. Bei kleinen Sektoren muß man mit einer kleinen Schleife mehrere Sektoren auf die Diskette schreiben, bis z. B. das Zeichen 0 im Sektor war, denn 0 bedeutet bei uns Ende des Textes. Jetzt kann man ins Menü Floppy-Start gehen und das Programm wieder aufrufen. Die Floppy greift zu und es muß eine senkrechte Linie erscheinen. Wenn man danach irgend eine Taste drückt, muß man wieder ins Grundprogramm zurückkehren.

Achtung: Der Aufruf von GRAPOFF ist hier nötig, da keine Rücksetzung der Schildkrötengrafik vorgenommen wurde.

```
start:
 nop     * Erkennung fuer Starter
 move #100,d0  * hier beliebiges Programm
 jsr @schreite * eingeben
 jsr @ci       * und dann Taste abwarten
 jsr @grapoff  * sonst bleibt Schildkroete
 rts           * das dann gestartet wird
speichern:     * Hilfsprogramm
 move #1,d4    * Laufwerk A
 jsr @getflop  * Dichte bestimmen
 lea start,a0  * Programmstart
 move #2,d1    * Speicher-Befehl
 move #1,d2    * Sektor 1
 move #0,d3    * Spur 0
 jsr @floppy   * in D4 Dichtecode
 lea $9000,a0  * Text auch speichern
 move #2,d1    * Speicherbefehl
 move #2,d2    * Auf Sektor 2
 move #0,d3    * Spur 0
 jsr @floppy   * und sichern
 rts

 end
```

Textstart=009000 Fenster=009000 Tor=009000 amer CTRL-J=Hilfe

Bild 7. Dieses Programm speichert sich selbst ab

```
1 = Speichern CAS
2 = Laden CAS
3 = Pruefen CAS
4 = Floppy Start
W = weiter
```

Bild 6. Das Menü mit dem Floppy-Start

```
lade:
 move #$1,d4     * Laufwerk A
 jsr @getflop    * Dichte bestimmen
 lea $a000,a0    * Text dorthin
 move #1,d1      * LESEN
 move #2,d2      * von Sektor 2
 move #0,d3      * Spur 0
 jsr @floppy     * und ausfuehren
 rts
```

Textstart=009000 Fenster=009000 Tor=009000 amer CTRL-J=Hilfe

Bild 8. Ein Sektor wird geladen

```
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

009C00                  ****************************************
009C00                  * MIKRODOS, (C) 1985 ROLF-DIETER KLEIN  *
009C00                  ****************************************
009C00                  * REV 3, QUELLE $A000 FIX, DIR AUF $9000
= 00000000              SDIR EQU 0 * KENNUNG
= 00000002              SNAM EQU 2 * NAME
= 0000000A              SFUL EQU 10 * DUMMY
= 0000000C              SSEK EQU 12 * ERSTE SEKTOR
= 0000000E              STRK EQU 14 * TRACK
009C00
009800                   ORG $9800  * PLATZ FUER QUELLE LASSEN
009800                  START:      * SPAETER AUF $9800 BIS $9BFF
009800  4E71            NOP * KENNUNG
009802  4241            CLR D1 *
009804  363C 0083       MOVE #$83,D3 * BEFEHL STEPRATE UND SSO
009808  3E3C 004B       MOVE #75,D7
00980C  4E41            TRAP #1 * FLOPPY
00980E  41FA 03B4       LEA MENUE(PC),A0
009812  6100 034E       BSR PRINT
009816  6100 0306       BSR CI
00981A  0C00 0031       CMP.B #'1',D0 * INH
00981E  6700 023B       BEQ INHALT
009822  0C00 0032       CMP.B #'2',D0 * LADEN
009826  6700 0134       BEQ LADEN
00982A  0C00 0033       CMP.B #'3',D0
00982E  6600 034A       BNE RETEX      * ENDE, SONST SPEICHERN
009832  6100 01E6       BSR GETNAM
009836  4A04            TST.B D4
009838  6700 0340       BEQ RETEX      * KEIN NAME DA
00983C                                 * SPEICHERN
00983C                                 * EINTRAGEN UND SPEICHERN
00983C                  * ERST DATEN
00983C  3439 00009B90    MOVE SEKTOR,D2
009842  3639 00009B92    MOVE TRACK,D3
009848  41F9 0000A000    LEA $A000,A0 * START
00984E  383C 0021        MOVE #$21,D4
009852                  SPELP:
009852  7202             MOVEQ #2,D1 * SCHREIBEN
009854  6100 02E4        BSR FLOPPY
009858  6500 0328        BCS ERROR
00985C  5242             ADDQ #1,D2 * NEUER SEKTOR
00985E  0C42 0006        CMP #6,D2 * 1..5
009862  660C             BNE.S SPE3CON
009864  7401             MOVEQ #1,D2
009866  5243             ADDQ #1,D3
009868  0C43 00A0        CMP #160,D3 * 0..159, DA DOPPELSEITIG
00986C  6700 0314        BEQ ERROR * IMMER WEITER SO.
009870                  SPE3CON:
009870  3A3C 03FF        MOVE #1024-1,D5 * ENDE SUCHEN
009874                  SPE2LP:
009874  4A18             TST.B (A0)+
009876  6700 000A        BEQ OKSENDE
00987A  51CD FFF8        DBRA D5,SPE2LP
00987E  6000 FFD2        BRA SPELP
009882                  OKSENDE:
009882  33C2 00009B94    MOVE D2,SEKNEU
009888  33C3 00009B96    MOVE D3,TRKNEU
00988E                  *
00988E  41F9 0000A000    LEA $A000,A0
009894  7201             MOVEQ #1,D1
009896  3439 00009B98    MOVE DIRSEK,D2
00989C  0C42 0006        CMP #6,D2      * FULL
0098A0  6700 02DA        BEQ FULERR
0098A4  4243             CLR D3     * DIREKTOR WAECHST JEDESMAL
0098A6  383C 0021        MOVE #$21,D4
0098AA  6100 028E        BSR FLOPPY     * DIR HOLEN, A0 BLEIBT
0098AE  6500 02D2        BCS ERROR
0098B2  3A39 00009B9A    MOVE DIRBYTE,D5
0098B8  4270 5000        CLR.W SDIR(A0,D5.W)
0098BC  21B9 00009BF0    MOVE.L BUFFER,SNAM(A0,D5.W)
0098C2  5002
0098C4  21B9 00009BF4    MOVE.L BUFFER+4,SNAM+4(A0,D5.W)
0098CA  5006
0098CC  4270 500A        CLR.W SFUL(A0,D5.W)
0098D0  31B9 00009B90    MOVE.W SEKTOR,SSEK(A0,D5.W)
0098D6  500C
0098D8  31B9 00009B92    MOVE.W TRACK,STRK(A0,D5.W)
0098DE  500E
0098E0  323C 0002        MOVE #2,D1
0098E4  3439 00009B98    MOVE DIRSEK,D2
0098EA  363C 0000        MOVE #0,D3
0098EE  383C 0021        MOVE #$21,D4
0098F2  6100 0246        BSR FLOPPY
0098F6  6500 028A        BCS ERROR
0098FA                  * ALTER SEKTOR ZURUECK DANN
0098FA                  * TEXTGEBIET WIEDER OK.
0098FA  7201             MOVEQ #1,D1
0098FC  3439 00009B90    MOVE SEKTOR,D2
009902  3639 00009B92    MOVE TRACK,D3
009908  6100 0230        BSR FLOPPY
00990C  6500 0274        BCS ERROR
009910                  *
009910  33F9 00009B94    MOVE SEKNEU,SEKTOR
009916  00009B90
00991A  33F9 00009B96    MOVE TRKNEU,TRACK
009920  00009B92
009924  0679 0010        ADD #16,DIRBYTE
009928  00009B9A
00992C  0C79 0400        CMP #1024,DIRBYTE
009930  00009B9A
009934  660C             BNE.S SPE1
009936  4279 00009B9A    CLR DIRBYTE
00993C  5279 00009B98    ADDQ #1,DIRSEK * NEUER SEKTOR
009942                  SPE1:
009942  41F9 00009800    LEA $9800,A0  * PROGRAMM NEU SPEICHERN
009948  7202             MOVEQ #2,D1
00994A  7401             MOVEQ #1,D2
00994C  4243             CLR D3
00994E  383C 0021        MOVE #$21,D4
009952  6100 01E6        BSR FLOPPY
009956  6500 022A        BCS ERROR
00995A  4E75             RTS
00995C
00995C
00995C                  LADEN: * SUCHEN NAMEN, DANN TRK,SEK LADEN
00995C  6100 00BC        BSR GETNAM
009960  4A04             TST.B D4
009962  6700 0216        BEQ RETEX * KEIN NAME DA
009966  7A03             MOVEQ #4-1,D5 * ANZAHL
009968  383C 0021        MOVE #$21,D4 * DD
00996C  7402             MOVEQ #2,D2 * SEKTOR 1
00996E  4243             CLR D3 * TRACK 0
009970                  LAD1:
009970  41F9 0000A000    LEA $A000,A0 * ZIELADRESSE
009976  7201             MOVEQ #1,D1 * LESEN
009978  6100 01C0        BSR FLOPPY * SPEZ.
00997C  6500 0204        BCS ERROR
009980  41F9 0000A000    LEA $A000,A0 * QUELLE
009986  3C3C 003F        MOVE #64-1,D6 * EINTRAEGE
00998A                  LAD2:
00998A  4A28 0000        TST.B 0(A0)  * GUELTIG ?
00998E  661A             BNE.S LAD3A
009990  202B 0002        MOVE.L SNAM(A0),D0
009994  B0B9 00009BF0    CMP.L BUFFER,D0
00999A  660E             BNE.S LAD3A
00999C  2028 0006        MOVE.L SNAM+4(A0),D0
0099A0  B0B9 00009BF4    CMP.L BUFFER+4,D0
0099A6  6700 0020        BEQ FOUND
0099AA                  LAD3A:
0099AA  D1FC 00000010    ADDA.L #16,A0 * NAECHSTER EINTRAG
0099B0  51CE FFD8        DBRA D6,LAD2
0099B4  5242             ADDQ #1,D2 * NEUER SEKTOR MAX 2..5
0099B6  51CD FFB8        DBRA D5,LAD1
0099BA  41FA 01E0        LEA NONAM(PC),A0
0099BE  6100 01A8        BSR ERRPRT * NICHT GEFUNDEN
0099C2  6100 015A        BSR CI
0099C6  4E75             RTS
0099C8
0099C8                  FOUND:  * GEFUNDEN
0099C8  3428 000C        MOVE SSEK(A0),D2
0099CC  3628 000E        MOVE STRK(A0),D3 * START
0099D0  383C 0021        MOVE #$21,D4 * DICHTE
0099D4  41F9 0000A000    LEA $A000,A0 * START DORT
0099DA                  FNDLP:
0099DA  7201             MOVEQ #1,D1
0099DC  6100 015C        BSR FLOPPY
0099E0  6500 01A0        BCS ERROR
0099E4  3A3C 03FF        MOVE #1024-1,D5 * ENDE SUCHEN
0099E8                  FND2LP:
0099E8  4A18             TST.B (A0)+
0099EA  6700 0020        BEQ OKENDE
0099EE  51CD FFF8        DBRA D5,FND2LP
0099F2  5242             ADDQ #1,D2 * NEUER SEKTOR
0099F4  0C42 0006        CMP #6,D2 * 1..5
0099F8  6600 FFE0        BNE FNDLP
0099FC  7401             MOVEQ #1,D2
0099FE  5243             ADDQ #1,D3
009A00  0C43 00A0        CMP #160,D3 * 0..159
009A04  6600 FFD4        BNE FNDLP * IMMER WEITER SO.
009A08  6000 0178        BRA ERROR * FATAL ERROR
009A0C                  OKENDE:
009A0C  203C 0000A000    MOVE.L #$A000,D0
009A12  3E3C 0043        MOVE #67,D7
```

Bild 9. Das Mikrodos im Listing

```
009A16  4E41            TRAP #1 * PUTSTX, NEUER TEXTSTART
009A18                  FINAL:
009A18  4E75            RTS
009A1A                  GETNAM:
009A1A  41FA 01C0       LEA NAME(PC),A0
009A1E  303C 0022       MOVE #$22,D0
009A22  323C 000A       MOVE #10,D1
009A26  343C 0064       MOVE #100,D2
009A2A  3E3C 000A       MOVE #10,D7
009A2E  4E41            TRAP #1 * NAME:
009A30  42B9 00009BF0   CLR.L BUFFER
009A36  42B9 00009BF4   CLR.L BUFFER+4
009A3C  41FA 01B2       LEA BUFFER(PC),A0 * ZIEL
009A40  303C 0022       MOVE #$22,D0
009A44  323C 0064       MOVE #100,D1
009A48  343C 0064       MOVE #100,D2
009A4C  363C 000B       MOVE #8,D3 * MAX 8 ZEICHEN
009A50  3E3C 000B       MOVE #11,D7
009A54  4E41            TRAP #1  * READ
009A56  4E75            RTS
009A58
009A58
009A58                  INHALT:
009A58  3E3C 0014       MOVE #20,D7
009A5C  4E41            TRAP #1 * CLRSCREEN
009A5E  7A03            MOVEQ #4-1,D5 * ANZAHL
009A60  383C 0021       MOVE #$21,D4 * DD
009A64  7402            MOVEQ #2,D2 * SEKTOR 1'
009A66  4243            CLR D3 * TRACK 0
009A68                  INH1:
009A68  41F9 00009000   LEA $9000,A0 * ZIELADRESSE
009A6E  7201            MOVEQ #1,D1 * LESEN
009A70  6100 00C8       BSR FLOPPY * SPEZ.
009A74  6500 010C       BCS ERROR
009A78  41F9 00009000   LEA $9000,A0 * QUELLE
009A7E  3C3C 003F       MOVE #64-1,D6 * EINTRAEGE
009A82                  INH2:
009A82  4A28 0000       TST.B 0(A0)  * GUELTIG ?
009A86  6622            BNE.S INH3A
009A88  48E7 0280       MOVEM.L D6/A0,-(A7)
009A8C  3C3C 0007       MOVE #8-1,D6
009A90  D1FC 00000002   ADDA.L #SNAM,A0
009A96                  INH3:
009A96  1018            MOVE.B (A0)+,D0
009A98  6708            BEQ.S INH3B  * 0= ENDE
009A9A  6100 0096       BSR CO2
009A9E  51CE FFF6       DBRA D6,INH3
009AA2                  INH3B:
009AA2  6100 00B2       BSR CRLF
009AA6  4CDF 0140       MOVEM.L (A7)+,D6/A0
009AAA                  INH3A:
009AAA  D1FC 00000010   ADDA.L #16,A0 * NAECHSTER EINTRAG
009AB0  51CE FFD0       DBRA D6,INH2
009AB4  5242            ADDQ #1,D2 * NEUER SEKTOR MAX 2..5
009AB6  51CD FFB0       DBRA D5,INH1
009ABA  6100 006A       BSR CRLF * REST INFO AUSGEBEN
009ABE  6100 0066       BSR CRLF
009AC2  6100 0062       BSR CRLF
009AC6  4240            CLR D0
009AC8  4241            CLR D1
009ACA  3E3C 001B       MOVE #27,D7 * NEWPAGE
009ACE  4E41            TRAP #1
009AD0  41FA 0110       LEA FREI(PC),A0
009AD4  6100 00B8       BSR PRINT2
009AD8  41FA 0116       LEA BUFFER(PC),A0
009ADC  3039 00009B92   MOVE TRACK,D0
009AE2  C0FC 0005       MULU #5,D0
009AE6  D079 00009B90   ADD SEKTOR,D0
009AEC  0640 FCE0       ADD #-800,D0 * XK FREI
009AF0  4440            NEG D0
009AF2  3E3C 002E       MOVE #46,D7
009AF6  4E41            TRAP #1 * PRINT4D
009AF8  41FA 00F6       LEA BUFFER(PC),A0
009AFC  6100 005B       BSR PRINT3
009B00  3E3C 0052       MOVE #82,D7
009B04  4E41            TRAP #1 * CURSORMODE AUS. CUROFF
009B06  4240            CLR D0
009B08  4241            CLR D1
009B0A  3E3C 0022       MOVE #34,D7
009B0E  4E41            TRAP #1 * SETFLIP AUF 0
009B10                  WARTE:
009B10  6100 000C       BSR CI  * DANN WARTEN, BIS TASTE
009B14  0C00 000D       CMP.B #$D,D0
009B18  66F6            BNE.S WARTE
009B1A  6000 FCE4       BRA START * DANN NOCHMALS FRAGEN
009B1E

009B1E                  CI:
009B1E  3E3C 000C       MOVE #12,D7
009B22  4E41            TRAP #1    * CI, AUSWAHL
009B24  4E75            RTS
009B26
009B26                  CRLF:
009B26  103C 000D       MOVE.B #$D,D0
009B2A  6100 0006       BSR CO2
009B2E  103C 000A       MOVE.B #$A,D0
009B32                  CO2:
009B32  3E3C 0021       MOVE #33,D7
009B36  4E41            TRAP #1  * CO2
009B38  4E75            RTS
009B3A
009B3A
009B3A                  FLOPPY:  * MIT TRACKUMRECHNUNG
009B3A  48E7 7E00       MOVEM.L D1-D6,-(A7)
009B3E  E25B            ROR #1,D3
009B40  6404            BCC.S FLO1
009B42  08C4 0007       BSET #7,D4 * RUECKSEITE
009B46                  FLO1:
009B46  0203 007F       AND.B #$7F,D3
009B4A  3E3C 004B       MOVE #75,D7
009B4E  4E41            TRAP #1 * FLOPPY
009B50  4CDF 007E       MOVEM.L (A7)+,D1-D6
009B54  4E75            RTS
009B56
009B56                  PRINT3:
009B56  323C 0052       MOVE #10+72,D1
009B5A  4242            CLR D2
009B5C  6012            BRA.S PRTA
009B5E                  PRINT2:
009B5E  4242            CLR D2
009B60  600A            BRA.S PRT
009B62                  PRINT:
009B62  343C 00C8       MOVE #200,D2
009B66  6004            BRA.S PRT
009B68                  ERRPRT:
009B68  343C 0014       MOVE #20,D2
009B6C                  PRT:
009B6C  323C 000A       MOVE #10,D1
009B70                  PRTA:
009B70  103C 0022       MOVE.B #$22,D0
009B74  3E3C 000A       MOVE #10,D7
009B78  4E41            TRAP #1  * WRITE
009B7A                  RETEX:
009B7A  4E75            RTS
009B7C
009B7C                  FULERR:
009B7C  41FA 003D       LEA DIRFUL(PC),A0
009B80  6004            BRA.S ERR1
009B82                  ERROR:
009B82  41FA 0021       LEA DOSERR(PC),A0
009B86                  ERR1:
009B86  6100 FFE0       BSR ERRPRT
009B8A  6100 FF92       BSR CI
009B8E  4E75            RTS
009B90
009B90                  * VARIABLE
009B90  0004            SEKTOR: DC.W 4 * FREIER SEKTOR
009B92  0005            TRACK: DC.W 5  * CP/M KOMP.
009B94  0004            SEKNEU: DC.W 4
009B96  0005            TRKNEU: DC.W 5
009B98  0002            DIRSEK: DC.W 2 * DIREKTORY
009B9A  0000            DIRBYTE: DC.W 0 * 0,16,...
009B9C                  * TEXTE
009B9C  6E696368742064  NONAM: DC.B 'nicht da',0
009BA3  6100
009BA5  4469736B204665  DOSERR: DC.B 'Disk Fehler',0
009BAC  686C657200
009BB1  4469736B20766F  DISKFUL: DC.B 'Disk voll',0
009BB8  6C6C00
009BBB  44697220766F6C  DIRFUL:  DC.B 'Dir voll',0
009BC2  6C00
009BC4  4469723D31204C  MENUE: DC.B 'Dir=1 Laden=2 Speich.=3',0
009BCB  6164656E3D3220
009BD2  5370656963682E
009BD9  3D3300
009BDC  4E616D653A00    NAME: DC.B 'Name:',0
009BE2  467265693A2020  FREI: DC.B 'Frei:      K',0
009BE9  202020204B00
009BEF  00               DS 0  * EVEN
009BF0                  BUFFER: DS.B 12 * AUF EVEN, WEGEN .L
009BFC
009BFC                   END

008E62  Ende-Symboltabelle
```

Ein gewagtes Manöver

Schalten Sie den Computer aus. Aber zuvor unbedingt die Diskette herausnehmen. Dann schalten Sie den Computer wieder ein. Diskette wieder einlegen und das Menü Floppy-Start anwählen. Dann muß wieder die senkrechte Linie erscheinen.

Ein kleines Ladeprogramm, das *Bild 8* zeigt, hilft dabei, den Text wiederzugewinnen. Wenn man es startet, so wird der Programmtext von Sektor 2 auf Spur 0 nach Adresse $a000 geholt. Man wähle nach dem Start von Lade das Menü TEXTSTART aus dem Grundprogramm und für den alten Text dann die Adresse $a000. Wenn man jetzt den Editor aufruft, so erhält man das vorherige Programm auf dem Bildschirm.

Hat man mehrere Sektoren zum Abspeichern eines Textes verwendet, so muß man beim Laden entsprechend mehrere Sektoren wieder einlesen.

Ein Betriebssystem wird benötigt

Mit den besprochenen Unterprogrammen kann man Texte und Programme einfach auf der Floppy abspeichern. Man müßte sich dazu nur auf einer Karteikarte den Namen eines jeden Programms und die Sektoren und Spuren notieren, die das Programm belegt. Einfacher geht das, wenn man dazu wiederum ein Programm verwendet, das diese Aufgabe automatisch ausführt. Das tut das sogenannte DOS, das Disk Operating System, das Diskettenbetriebsystem.

Ein einfaches DOS wollen wir uns nun einmal ansehen. *Bild 9* zeigt das Listing eines solchen Programmes, das schon ganz gut brauchbar ist.

Es wird ab Adresse $A000 eingegeben, das ist sehr wichtig. Das Programm

Bild 10. Das Menü von Mikrodos ist simpel

Bild 11. Das Speichern von Mikrodos selbst

selbst muß auf Adresse $9800 gelegt werden, die Adresse, auf die es später auch vom Grundprogramm beim Floppy-Start geladen wird. Nach der Übersetzung des Programms wird der Name „START" aufgerufen, es meldet sich dann ein Menü, wie es *Bild 10* zeigt. Mit der Funktion 3 kann man jetzt das Programm selbst abspeichern. Nach Tippen von 3 ergibt sich *Bild 11*. Im Fenster gibt man den Namen „dos" ein. Wenn man jetzt die Taste CR betätigt, so wird das Programm sowohl als Text, als auch für Adresse $9800 assembliert, in den ersten Sektor geschrieben. Damit ist unser kleines DOS betriebsbereit.

Achtung: Das Listing sollten Sie sehr sorgfältig kontrollieren, bevor Sie das Programm starten, ein Fehler könnte Ihnen sonst später einmal viel Arbeit zunichte machen. Das DOS arbeitet mit dem NDR80-Format, um maximale Kapazität zu erreichen. Wer andere Laufwerke besitzt, muß die Parameter entsprechend ändern. Dabei muß aber immer ein Format mit 1024 Bytes pro Sektor verwendet werden, das ECMA70-Format ist also nicht verwendbar. Das DOS arbeitet wie folgt. Auf den Sektoren 2 bis 5 der Spur 0 einer jeden Diskette ist das Inhaltsverzeichnis untergebracht, das auch Directory genannt wird. Jeder Sektor kann dabei 64 Dateinamen aufnehmen. *Bild 12* zeigt den Aufbau eines Directoryblocks. Das erste Wort dient zu Kennung, nämlich ob der Eintrag als gelöscht gilt oder nicht. Nach dem Formatieren einer Diskette steht dort der Wert

Bild 13. Das Inhaltsverzeichnis unter Mikrodos

$E5E5. Wenn eine Datei eingetragen wird, so wird dort ein anderer Wert eingeschrieben. Dann folgt der Dateiname, der maximal aus 8 Zeichen bestehen kann. Die Bytes 10 und 11 sind mit 0 belegt und können z. B. für Erweiterungen des Systems verwendet werden.

Dann folgt der Startsektor des Eintrags und dann die Startspur der Datei. Eine Datei soll aufhören, wenn in einem Sektor der Wert 0 vorkommt (so muß man sich hier den Endesektor nicht merken). Mit dieser Technik lassen sich aber nur Textdateien abspeichern. Die Größen Sektor und Spur sind bei anderen Betriebssystemen normalerweise nur je 1 Byte lang.

Die Aufgaben von DOS

Unser DOS hat nur drei verschiedene Aufgaben. Zunächst zur Verwaltung des Inhaltsverzeichnisses. *Bild 13* zeigt die Ausgabe eines Inhaltsverzeichnisses. Unter DOS werden alle Directorysektoren einer Diskette dazu eingelesen, und es wird nur jeweils der Namenseintrag ausgegeben, bei dem es sich um einen gültigen Eintrag handelt. Am Schluß des Programms wird noch der verbleibende Platz auf der Diskette ausgegeben, der sich aus dem aktuellen Sektor ergibt.

Das ist der, der als nächstes verwendet werden kann. Dessen Spurnummer multipliziert man mit 5 und addiert dazu die Nummer des Sektors. Man erhält dadurch die Anzahl der belegten Bytes,

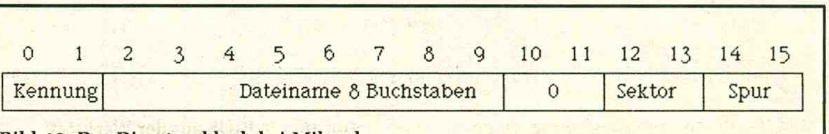


Bild 12. Der Directoryblock bei Mikrodos

denn unser DOS speichert zunächst nur fortlaufend. Den freien Platz erhält man, indem man die Gesamtkapazität, also 800 KByte, hernimmt und davon den eben errechneten Wert subtrahiert.

Beim Laden einer Datei, der zweiten Aufgabe von DOS, wird zunächst aus Directory nach dem Namen der gewünschten Datei durchsucht. Ist der Name nicht auffindbar, so erscheint eine Fehlermeldung (*Bild 14*). Der Name „ha" wurde nicht gefunden. Beim Vergleich werden übrigens alle eingetippten Zeichen auch ausgewertet, auch die Leertastenanschläge. Also immer ganz genau tippen – sonst funktioniert es nicht. Ist der Name in Directory aufgefunden, dann wird ab der angegebenen Spur/Sektor-Nummer solange eingelesen, bis ein Byte entdeckt wird, das Null ist. Jetzt die Funktion Speichern, die die komplizierteste ist. Zum einen muß der Name der neuen Datei im Inhaltsverzeichnis eingetragen werden. Dazu gibt es im Programm die Merker DIRSEK und DIRBYTE. Doch zuvor wird erst der Text gespeichert, der hier auf Adresse $A000 beginnen soll, denn das DOS sitzt ja auf Adresse $9800 und der Platz zwischen $9000 und $9800 wäre zu knapp für Texte. Sie müssen also TEXTSTART= $A000 einstellen, ehe Sie mit DOS arbeiten können. Der erste freie Sektor steht in der Variablen SEKTOR und die erste freie Spur in der Variablen TRACK. Es werden nun soviele Sektoren abgespeichert, bis der Wert 0 im Sektor auftrat. Dann wird der neue Wert des Sektors in NEUSEK und die neue Spur in NEUTRK abgespeichert. Jetzt wird das Inhaltsverzeichnis geladen und zwar bei dem Sektor DIRSEK, dem letzten belegten Sektor.

```
Dir=1 Laden=2 Speich.=3

Name:  ha

nicht da
```

Bild 14. Wenn eine Datei nicht da ist

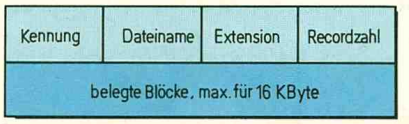


Bild 15. Der Aufbau eines CP/M-Directory-Blocks

DIRBYTE gibt den aktuellen Directoryplatz im Sektor an. Dorthin wird der Name des Programms gespeichert, wie auch der Inhalt der Variablen SEKTOR und TRACK, die ja noch den Start der Daten angeben. Danach wird der Directorysektor zurückgespeichert. Dann wird der nächste Directoryblock berechnet, also zunächst DIRBYTE + 16, denn 16 Bytes belegt ein Block. Falls der Sektor voll ist, wird der nächste Sektor angewählt und DIRBYTE wieder auf 0 gesetzt. Man kann bis zu 256 Einträge durchführen, dann erscheint die Fehlermeldung DIR voll oder die Fehlermeldung Disk voll (wenn man zuviele Daten unterbringen wollte).

Am Schluß wird, und das ist der Angelpunkt der Sache, der Bereich $9800 bis $9BFF auf den Sektor 1, Spur 0 geschrieben. Denn damit werden alle Variablen und das DOS-System selbst automatisch gerettet. Das war auch der Grund, weshalb bei der Inbetriebnahme zunächst die Funktion SPEICHERN aufgerufen werden mußte.

Wenn man weitere Disketten vorbereiten will, so muß man das DOS mit LADEN wieder zurückholen. Man muß es dann erneut übersetzen, damit alle Variablen neu initialisiert werden. Dann kann man eine neue formatierte Diskette einlegen und das Programm wie besprochen wieder abspeichern.

In Zukunft braucht man dann nur noch FLOPPY-Start eingeben, und das Grundprogramm holt den Sektor und startet unser Mikrodos.

Mikrodos ist natürlich sehr klein, viel Komfort fehlt noch. Aber man kann Programme speichern und laden, und das ist ja auch schon etwas. Man könnte sich aber Hilfsprogramme auf Diskette ablegen, die z. B. einen Eintrag löschen oder eine Diskette kopieren.

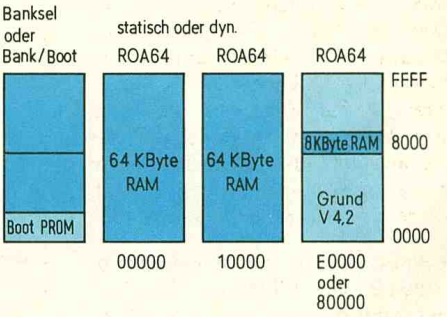


Bild 17. Die Speicheraufteilung für CP/M-68k

```
...A68K 4.0a 06/13/83   ...Run on 01/09/85
                          1. ******************************************
                          2. * BIOS 68k 1.2 fuer  NDR-Klein-Computer  *
                          3. * Version 1.2  B41007 R-D.Klein          *
                          4. ******************************************
                          5.
                          6. * mit IMAGE laden als EPROM
                          7. * startwarm ccp bei $15000
      000150B8            8. ccp equ $150b8
      00019F84            9. patch equ $19f84
                         10.
                         11.
      000B0000           12. gru68k equ $B0000  * spaeter E0000, verschiebbar
      000A0000           13. cpm equ $a0000
      00015000           14. dest equ $15000 * Ziel
                         15.
                         16. * Version Gru68k auf 80000
                         17. * CPM + BIOS auf a0000
                         18.
      0001B000           19. basis equ $1b000
                         20.
```

Bild 16. Ein BIOS aus der Werkstatt

```
01B000                  21.  org.l $1b000   * Start Bios
                        22.
01B000                  23. init:
01B000  4279 0001B250   24.   clr cursor    * Cursor ist jetzt aus
01B006  21FC 0001B012   25.   move.l #traphndl,$8c *trap #3 handler
        008C
01B00E  4280            26.   clr.l d0
01B010  4E75            27.   rts
                        28.
      00000004          29. maxdsk equ 4
      0000001A          30. dphlen equ 26
                        31.
01B012                  32. traphndl:
01B012  0C40 0018       33.  cmpi #nfuncs,d0
01B016  6400 0012       34.  bcc trapng
01B01A  E548            35.  lsl #2,d0
01B01C  48C0            36.  ext.l d0
01B01E  D0BC 0001B02C   37.  add.l #biosbase,d0
01B024  2040            38.  movea.l d0,a0
01B026  2050            39.  movea.l (a0),a0
01B028  4E90            40.  jsr (a0)
01B02A                  41. trapng:
01B02A  4E73            42.  rte
                        43.
01B02C                  44. biosbase:
01B02C  0001B000        45.  dc.l init      * 0
01B030  0001B094        46.  dc.l wboot     * 1
01B034  0001B09A        47.  dc.l constat   * 2
01B038  0001B0CE        48.  dc.l conin     * 3
01B03C  0001B0F6        49.  dc.l conout    * 4
01B040  0001B11A        50.  dc.l lstout    * 5
01B044  0001B126        51.  dc.l pun       * 6
01B048  0001B132        52.  dc.l rdr       * 7
01B04C  0001B142        53.  dc.l home      * 8
01B050  0001B14A        54.  dc.l seldsk    * 9
01B054  0001B166        55.  dc.l settrk    * 10
01B058  0001B16E        56.  dc.l setsec    * 11
01B05C  0001B182        57.  dc.l setdma    * 12
01B060  0001B1C6        58.  dc.l read      * 13
01B064  0001B1F6        59.  dc.l write     * 14
01B068  0001B13C        60.  dc.l listst    * 15
01B06C  0001B176        61.  dc.l sectran   * 16
01B070  0001B182        62.  dc.l setdma    * 17
01B074  0001B22C        63.  dc.l getseg    * 18
01B078  0001B234        64.  dc.l getiob    * 19
01B07C  0001B236        65.  dc.l setiob    * 20
01B080  0001B228        66.  dc.l flush     * 21
01B084  0001B238        67.  dc.l setexc    * 22
01B088  0001B08C        68.  dc.l getbasis  * 23   neuer Bios-Einqang. a4 belegen
                                                   mit BASIS
                        69.
      00000018          70. nfuncs equ (*-biosbase)/4
                        71.
01B08C                  72. getbasis:
01B08C  49F9 00080000   73.  lea gru68k,a4
01B092  4E75            74.  rts
                        75.
01B094                  76. wboot:
01B094  4EF9 000150B8   77.  jmp ccp
                        78.
01B09A                  79. constat:
01B09A  4A79 0001B250   80.  tst cursor
```

```
01B0A0  6600 0012       81.  bne conlstat   * Cursor ein, dann skip
01B0A4  33FC 0001       82.  move #1,cursor
        0001B250
01B0AC  3E3C 003D       83.  move #61,d7
01B0B0  6100 0196       84.  bsr trapexe    * CURSEIN
01B0B4                  85. conlstat:
01B0B4  3E3C 003C       86.  move #60,d7
01B0B8  6100 018E       87.  bsr trapexe    * Autoflip
01B0BC  3E3C 000D       88.  move #13,d7
01B0C0  6100 0186       89.  bsr trapexe    * CSTS
01B0C4  6704            90.  beq.s noton
01B0C6  7001            91.  moveq.l #$1,d0
01B0C8  4E75            92.  rts
01B0CA                  93. noton:
01B0CA  4280            94.  clr.l d0
01B0CC  4E75            95.  rts
                        96.
01B0CE                  97. conin:
01B0CE  4A79 0001B250   98.  tst cursor
01B0D4  6700 0010       99.  beq conlin
01B0D8  4279 0001B250  100.  clr cursor
01B0DE  3E3C 003E      101.  move #62,d7
01B0E2  6100 0164      102.  bsr trapexe    * CURSAUS
01B0E6                 103. conlin:
01B0E6  3E3C 000C      104.  move #12,d7
01B0EA  6100 015C      105.  bsr trapexe
01B0EE  C0BC 0000007F  106.  and.l #$7f,d0  * Wort wird ausgewertet
01B0F4  4E75           107.  rts
                       108.
01B0F6                 109. conout:
01B0F6  4A79 0001B250  110.  tst cursor
01B0FC  6700 0010      111.  beq conlout
01B100  4279 0001B250  112.  clr cursor
01B106  3E3C 003E      113.  move #62,d7
01B10A  6100 013C      114.  bsr trapexe    * CURSAUS
01B10E                 115. conlout:
01B10E  3001           116.  move d1,d0
01B110  3E3C 0021      117.  move #33,d7
01B114  6100 0132      118.  bsr trapexe
01B118  4E75           119.  rts
                       120.
01B11A                 121. lstout:
01B11A  3001           122.  move d1,d0
01B11C  3E3C 0016      123.  move #22,d7
01B120  6100 0126      124.  bsr trapexe
01B124  4E75           125.  rts
                       126.
01B126                 127. pun:
01B126  3001           128.  move d1,d0
01B128  3E3C 000F      129.  move #15,d7
01B12C  6100 011A      130.  bsr trapexe
01B130  4E75           131.  rts
                       132.
01B132                 133. rdr:
01B132  3E3C 000E      134.  move #14,d7
01B136  6100 0110      135.  bsr trapexe
01B13A  4E75           136.  rts
                       137.
01B13C                 138. listst:
01B13C  103C 00FF      139.  move.b #$ff,d0
01B140  4E75           140.  rts
                       141.
```

```
01B142                  142. home:
01B142  4239 0001B256   143.  clr.b track
01B148  4E75            144.  rts
                        145.
01B14A                  146. seldsk:
01B14A  7000            147.  moveq #0,d0
01B14C  B23C 0004       148.  cmp.b #maxdsk,d1        * max disk
01B150  6A00 0012       149.  bpl selrtn
01B154  13C1 0001B258   150.  move.b d1,seldrv
01B15A  C0FC 001A       151.  mulu #dphlen,d0
01B15E  D0BC 0001B264   152.  add.l #dph0,d0
01B164                  153. selrtn:
01B164  4E75            154.  rts
                        155.
01B166                  156. settrk:
01B166  13C1 0001B256   157.  move.b d1,track
01B16C  4E75            158.  rts
                        159.
01B16E                  160. setsec:
01B16E  13C1 0001B257   161.  move.b d1,sector
01B174  4E75            162.  rts
                        163.
01B176                  164. sectran:
01B176  2042            165.  movea.l d2,a0
01B178  48C1            166.  ext.l d1
01B17A  1030 1000       167.  move.b 0(a0,d1),d0
01B17E  48C0            168.  ext.l d0
01B180  4E75            169.  rts
                        170.
01B182                  171. setdma:
01B182  23C1 0001B252   172.  move.l d1,dma
01B188  4E75            173.  rts
                        174.
01B18A                  175. dcode:          * Codieren Drive + Dense
01B18A  1839 0001B258   176.  move.b seldrv,d4 * 0,1,2,3 ->
01B190  B83C 0000       177.  cmp.b #0,d4
01B194  6606            178.  bne.s d1code
01B196  183C 0011       179.  move.b #$11,d4          * A
01B19A  4E75            180.  rts
01B19C                  181. d1code:
01B19C  B83C 0001       182.  cmp.b #1,d4
01B1A0  6606            183.  bne.s d2code
01B1A2  183C 0012       184.  move.b #$12,d4          * B
01B1A6  4E75            185.  rts
01B1A8                  186. d2code:
01B1A8  B83C 0002       187.  cmp.b #2,d4
01B1AC  6606            188.  bne.s d3code
01B1AE  183C 0091       189.  move.b #$91,d4          * C
01B1B2  4E75            190.  rts
01B1B4                  191. d3code:
01B1B4  B83C 0003       192.  cmp.b #3,d4
01B1B8  6606            193.  bne.s d4code
01B1BA  183C 0092       194.  move.b #$92,d4          * D
01B1BE  4E75            195.  rts
01B1C0                  196. d4code:
01B1C0  183C 0011       197.  move.b #$11,d4 * ERROR
01B1C4  4E75            198.  rts
                        199.
                        200.
01B1C6                  201. read:
01B1C6  323C 0001       202.  move #1,d1      * Lesen
01B1CA  4242            203.  clr d2
01B1CC  4243            204.  clr d3

01B1CE  1439 0001B257   205.  move.b sector,d2
01B1D4  1639 0001B256   206.  move.b track,d3
01B1DA  61AE            207.  bsr dcode
01B1DC  2079 0001B252   208.  movea.l dma,a0
01B1E2  3E3C 004B       209.  move #75,d7
01B1E6  6100 0060       210.  bsr trapexe
01B1EA  6706            211.  beq.s rdok
01B1EC  303C 0001       212.  move.w #1,d0
01B1F0  4E75            213.  rts
01B1F2                  214. rdok:
01B1F2  4240            215.  clr.w d0
01B1F4  4E75            216.  rts
                        217.
                        218.
                        219.
01B1F6                  220. write:
01B1F6  323C 0002       221.  move #2,d1      * Schreiben
01B1FA  4242            222.  clr d2
01B1FC  4243            223.  clr d3
01B1FE  1439 0001B257   224.  move.b sector,d2
01B204  1639 0001B256   225.  move.b track,d3
01B20A  6100 FF7E       226.  bsr dcode
01B20E  2079 0001B252   227.  movea.l dma,a0
01B214  3E3C 004B       228.  move #75,d7
01B218  6100 002E       229.  bsr trapexe
01B21C  6706            230.  beq.s wrok
01B21E  303C 0001       231.  move.w #1,d0
01B222  4E75            232.  rts
01B224                  233. wrok:
01B224  4240            234.  clr.w d0
01B226  4E75            235.  rts
                        236.
                        237.
01B228                  238. flush:
01B228  4280            239.  clr.l d0
01B22A  4E75            240.  rts
                        241.
01B22C                  242. getseg:
01B22C  203C 0001B25A   243.  move.l #memrgn,d0
01B232  4E75            244.  rts
                        245.
01B234                  246. getiob:
01B234  4E75            247.  rts
                        248.
01B236                  249. setiob:
01B236  4E75            250.  rts
                        251.
01B238                  252. setexc:
01B238  0281 000000FF   253.  andi.l #$ff,d1
01B23E  E549            254.  lsl #2,d1
01B240  2041            255.  movea.l d1,a0
01B242  2010            256.  move.l (a0),d0
01B244  2082            257.  move.l d2,(a0)
01B246                  258. noset:
01B246  4E75            259.  rts
                        260.
01B248                  261. trapexe:        * Grundprog ausfuehren ueber Sprung
01B248  4EB9 00080420   262.  jsr gru68k+$420 * direkter Sprung moeglich rev 4.2
01B24E  4E75            263.  rts            * Flags ok.
                        264.
01B250  0000            265. cursor: dc.w 0  * Merker Cursor 1=ein, 0=aus
                        266.
01B252  00000000        267. dma: dc.l 0
```

Bemerkungen zu CP/M-68k

Nach dem Eigenbau-DOS hier noch die Beschreibung eines kommerziellen DOS, das sehr komfortabel, aber auch teuer ist. Es handelt sich um das Betriebssystem CP/M-68k.
Zunächst unterscheidet sich CP/M in der Art und Weise, wie es die Dateien auf der Diskette verteilt. Im Mikrodos wurde nur der Startsektor einer Datei angegeben. Wenn man eine Datei löschen will, und den Platz wieder gewinnen will, so müßte man daher alle nachfolgenden Dateien verschieben – und das ist sehr aufwendig. Im CP/M wird das Problem elegant umgangen. Man schreibt in den Directoryblock eines Eintrages Blocknummern, wobei jeder Block einer bestimmten Anzahl von Sektoren entspricht. 16 Blöcke bringt man dabei in einem Directoryblock unter (*Bild 15*). Wenn 16 Blöcke belegt sind, so wird von CP/M-68k automatisch eine neue Datei mit dem gleichen Namen, aber einer neuen Extensionnummer angelegt. So weiß man später genau, welche Blöcke

```
01B256 00          track: dc.b 0
01B257 00          sector: dc.b 0
01B258 00          seldrv: dc.b 0
01B259 00                  dc.b 0

01B25A 0001        memrgn: dc.w 1
01B25C 00000400            dc.l $400
01B260 00014C00            dc.l $14c00

                   * disk parameters

01B264 0001B2DC    dph0:   dc.l xlt
01B268 0000                dc.w 0
01B26A 0000                dc.w 0
01B26C 0000                dc.w 0
01B26E 0001B2F6            dc.l dirbuf
01B272 0001B2CC            dc.l dpb
01B276 0001B376            dc.l ckv0
01B27A 0001B3B6            dc.l alv0

01B27E 0001B2DC    dph1:   dc.l xlt
01B282 0000                dc.w 0
01B284 0000                dc.w 0
01B286 0000                dc.w 0
01B288 0001B2F6            dc.l dirbuf
01B28C 0001B2CC            dc.l dpb
01B290 0001B386            dc.l ckv1
01B294 0001B3D6            dc.l alv1

01B298             dph2:
01B298 0001B2DC            dc.l xlt
01B29C 0000                dc.w 0
01B29E 0000                dc.w 0
01B2A0 0000                dc.w 0
01B2A2 0001B2F6            dc.l dirbüf
01B2A6 0001B2CC            dc.l dpb
01B2AA 0001B396            dc.l ckv2
01B2AE 0001B3F6            dc.l alv2

01B2B2             dph3:
01B2B2 0001B2DC            dc.l xlt
01B2B6 0000                dc.w 0
01B2B8 0000                dc.w 0
01B2BA 0000                dc.w 0
01B2BC 0001B2F6            dc.l dirbuf
01B2C0 0001B2CC            dc.l dpb
01B2C4 0001B3A6            dc.l ckv3
01B2C8 0001B416            dc.l alv3

01B2CC             dpb:
01B2CC 001A                dc.w 26
01B2CE 03                  dc.b 3
01B2CF 07                  dc.b 7
01B2D0 00                  dc.b 0
01B2D1 00                  dc.b 0
01B2D2 00F2                dc.w 242
01B2D4 003F                dc.w 63
01B2D6 C000                dc.w $c000
01B2D8 0010                dc.w 16
01B2DA 0002                dc.w 2

01B2DC 01 07 0D 13 xlt:    dc.b 1,7,13,19
01B2E0 19 05 0B 11         dc.b 25,5,11,17
01B2E4 17 03 09 0F         dc.b 23,3,9,15
01B2E8 15 02 08 0E         dc.b 21,2,8,14
01B2EC 14 1A 06 0C         dc.b 20,26,6,12
01B2F0 12 18 04 0A         dc.b 18,24,4,10
01B2F4 10 16               dc.b 16,22

01B2F6             dirbuf:
01B2F6                     ds.b 128
01B376             ckv0:   ds.b 16
01B386             ckv1:   ds.b 16
01B396             ckv2:   ds.b 16
01B3A6             ckv3:   ds.b 16
01B3B6             alv0:   ds.b 32
01B3D6             alv1:   ds.b 32
01B3F6             alv2:   ds.b 32
01B416             alv3:   ds.b 32

                   * Temp COPY Programm

01B800             org basis+$800
01B800             ant:
01B800 55AA0180            dc.l $55aa0180
01B804 43 50 4D 20         dc.b 'CPM BOOT'
       42 4F 4F 54
01B80C 0000002C            dc.l ausf-anf
01B810 00000400            dc.l $400
01B814 01 00 00 00         dc.b 1,0,0,0
01B818 00000000            dc.l 0,0,0
       00000000
       00000000
01B824 00000000            dc.l 0,0
       00000000

01B82C             ausf:
01B82C 3E3C 0014           move #20,d7
01B830 4E41                trap #1
01B832 303C 0011           move #$11,D0
01B836 3E3C 0019           move #25,d7
01B83A 4E41                trap #1
01B83C 4241                clr d1
01B83E 4243                clr d3
01B840 3E3C 004B           move #75,d7
01B844 4E41                trap #1
01B846 41F9 00015000       lea $15000,a0
01B84C 43F9 000A0000       lea $a0000,a1
01B852 243C 000067FF       move.l #anf-$15000-1,d2
01B858             lpp:
01B858 10D9                move.b (a1)+,(a0)+
01B85A 51CA FFFC           dbra d2,lpp
01B85E 4EF9 00015000       jmp $15000

01B864             end

                   0 Errors detected
```

zu einer Datei gehören. Eine Rekordzahl in Directory-Eintrag gibt in 128-Byte-Schritten an, wieviel Platz genau belegt ist. Eine Kennung (E5) am Anfang sagt, ob ein Eintrag gültig ist. Ein solcher Eintrag belegt 32 Bytes, so daß man für 64 Einträge 2048 Bytes benötigt. Wenn man eine Datei löscht, so werden alle davon belegten Blöcke wieder verfügbar. Dazu wird eine sogenannte Allocate-Map (deutsch = Belegungstabelle) geführt. Es ist eine Bittabelle, in der je ein gesetztes Bit einem belegten Block entspricht. Dadurch , daß die Blöcke möglichst groß gewählt werden, minimal 1 KByte, meist 2 K oder 4 K, ist die Anzahl der Organisations-Bits nicht mehr so groß, sie hängt von der Diskettenkapazität ab. CP/M arbeitet mit Sektoren, die 128 Bytes lang sind. Wenn man Disketten mit einem anderen Format verwendet, so muß man die Sektoren entsprechend aufteilen.

CP/M ist anpaßbar

CP/M-68k ist an die verschiedensten Computer anpaßbar. Das geschieht mit einem sogenannten BIOS. *Bild 16* zeigt ein BIOS aus unserer Werkstatt, das für 8-Zoll-Disketten geschrieben wurde und hier nur als Beispiel dienen soll.

CP/M und NDR-68008

Wie muß der NDR-Klein-Computer aussehen, damit er CP/M-68k fahren kann? *Bild 17* zeigt ein Schema. Zunächst einmal benötigt man sehr viel RAM, nämlich 128 KByte Minimum. Die BANK/BOOT-Karte (oder die ältere BANKSEL-Karte) wird ebenfalls benötigt, da auch hier der RAM-Bereich bei Adresse 0 beginnen muß. Das Grundprogramm der Version 4.2 ist zum Glück verschiebbar, es kann auf einer getrennten ROA64-Baugruppe untergebracht werden. Es benötigt ein getrenntes 8-KByte-RAM auf der ROA an gewohnter Stelle, um dort den Bildwiederholspeicher unterzubringen. Man kann das Grundprogramm auf Adresse $80000 unterbringen, wenn man das „Werkstattbios" verwenden will oder auf Adresse $E0000, wenn man die endgültige Version verwendet, die es auf Floppy zu kaufen gibt. Auf der BOOT-Karte ist ein EPROM untergebracht, das die Aufgabe hat, das Grundprogramm zu starten und dann die BOOT-Karte wegzuschalten. Es wird dann nicht mehr angesprochen. Das Programm ist sehr kurz, *Bild 18* zeigt das Listing. Dabei sucht das BOOT-Programm selbständig das Grundprogramm und startet es, wenn gefunden. Auf

```
...A68K 4.0a 06/13/83   ...Run on 01/09/85
                         1. ************************************
                         2. * Boot fuer Bankboot/ Boot-Karten  *
                         3. * (C) 1984 Rolf-Dieter Klein       *
                         4. * V1.0 841016                     *
                         5. ************************************
                         6. * 68008 Boot
                         7.
000000                   8.          section 0
                         9.
                        10.
000000  00008FFE        11.          dc.l $8ffe       * dummy Stackpointer
000004  00000008        12.          dc.l start       * Start 1. Stufe
                        13.
                        14.
000008  363C 0031      15. start:   move #ende-anf-1,d3
00000C  41F9 00008000  16.          lea $8000,a0     * Ziel-Gebiet fuer Kalt-Start
000012  43FA 000E      17.          lea anf(pc),a1   * Quelle bei Start
000016                 18. transport:
000016  10D9           19.          move.b (a1)+,(a0)+
000018  51CB FFFC      20.          dbra d3,transport
00001C  4EF9 00008000  21.          jmp $8000        * Und ins Zielspringen
                        22.
                        23.
000022                 24. anf:
                        25.                          * Suchen nach ROM-Gebiet
                        26.                          * auf Adresse 0
000022  1038 7000      27. boot:    move.b $7000,d0 * Loeschen Boot
000026  11FC 00B0      28.          move.b #$80,$ffffffc8  * loeschen Bankboot
        FFC8
00002C  41F8 0400      29.          lea $400,a0      * Start der Suche
000030  0C90 5AA58001  30. loop:    cmp.l #$5aa58001,(a0)
000036  6708           31.          beq.s gefunden
000038  D1FC 00001000  32.          adda.l #$1000,a0 * Alle 4K Suchen nach dem Anfang
00003E  60F0           33.          bra.s loop
000040                 34. gefunden:
000040  0C68 6000      35.          cmp #$6000,$20(a0)
        0020
000046  66E8           36.          bne.s loop           * dort muss Sprung stehen
000048  0C68 6000      37.          cmp #$6000,$24(a0)   * dort muss Sprung stehen
        0024
00004E  66E0           38.          bne.s loop           * ok Test scheint ok
000050  4EE8 0024      39.          jmp $24(a0)          * also Cold-Start ausfuehren
000054                 40. ende:
                        41.
000054                 42.          end
    0 Errors detected
```

Bild 18. Das BOOT-Programm für den CP/M-68k-Start

```
C P / M   6 8 0 0 0   A s s e m b l e r                Revision 04.03
Source File: editrdk.s

  1                          ************************************************
  2                          * Disk-Editor mit Grundprogrammunterstuetzung *
  3                          * (C) 1984 Rolf-Dieter Klein   841021  V1.0   *
  4                          ************************************************
  5
  6                          * Grundprogrammaufruf mit Hilfe Funktion 23
  7                          * im BIOS
  8                          * Ladeadresse ist $9000
  9
 10                          ladeadr equ     $9000
 11
 12                          reboot  equ     0
 13                          open    equ     15
 14                          setdma  equ     26
 15                          make    equ     22
 16                          read    equ     20
 17                          write   equ     21
 18                          close   equ     16
 19                          printstr equ    9
 20
 21 00000000                  .text
 22
 23
 24                          start:
 25 00000000 4E560000         link a6,#0              * mark stack frame
 26 00000004 206E0008         move.l 8(a6),a0         * Adresse BASIS-Seite
 27 00000008 43E8005C         lea $5c(a0),a1          * FCB
```

Bild 19. Ein Diskeditor

```
28 0000000C 2209                move.l a1,d1            * Adresse FCB laden
29 0000000E 23C100000000        move.l d1,fcbadr        * dort merken
30 00000014 303C000F            move.w #open,d0
31 00000018 4E42                trap #2                 * OPEN
32 0000001A B07C00FF            cmp #255,d0             * =255 dann Datei nicht da
33 0000001E 661C                bne okda                * dann kreieren
34 00000020 223900000000        move.l fcbadr,d1
35 00000026 303C0016            move.w #make,d0
36 0000002A 4E42                trap #2                 * anlegen
37 0000002C 0C4000FF            cmpi #255,d0
38 00000030 670000F0            beq error               * Datei noch leer.
39 00000034 423900009000        clr.b ladeadr           * kennzeichnen
40 0000003A 6036                bra eingelesen
41
42                              okda:                   * Datei ist schon da
43                                                      * erst mal einlesen
44 0000003C 223C00009000        move.l #ladeadr,d1      * Startadresse
45                              schleife:
46 00000042 48E74000            movem.l d1,-(a7)
47 00000046 701A                move #setdma,d0
48 00000048 4E42                trap #2                 * DMA setzen
49 0000004A 223900000000        move.l fcbadr,d1
50 00000050 7014                move #read,d0
51 00000052 4E42                trap #2                 * LESEN
52 00000054 4CDF0002            movem.l (a7)+,d1
53 00000058 D2BC00000080        add.l #128,d1           * neue Adresse
54 0000005E 4A40                tst d0
55 00000060 67E0                beq schleife            * ok eingelesen
56 00000062 41F900009000        lea ladeadr,a0          * Suchen Nach 1A und ersetzen durch
57                              lpp:
58 00000068 0C18001A            cmp.b #$1a,(a0)+
59 0000006C 66FA                bne lpp
60 0000006E 4228FFFF            clr.b -1(a0)            * loeschen vorhergehendes Bytes
61                              eingelesen:             * Schliessen, Aufruf des Editors
62 00000072 223900000000        move.l fcbadr,d1
63 00000078 7010                move #close,d0
64 0000007A 4E42                trap #2                 * Datei schliessen
65                              ********************************************************
66                              * Editor Grundprogramm aufrufen
67                              *
68 0000007C 7017                move #23,d0
69 0000007E 4E43                trap #3                 * Sonderfunktion
70 00000080 23CC00000004        move.l a4,gruadr
71                              *
72 00000086 203C00009000        move.l #ladeadr,d0
73 0000008C 7E43                move #67,d7
74 0000008E 4EAC0420            jsr $420(a4)            * PUTSTX
75 00000092 2B7900000004        movea.l gruadr,a4
76 00000098 7E38                move #56,d7
77 0000009A 4EAC0420            jsr $420(a4)            * EDIT
78 0000009E 2B7900000004        movea.l gruadr,a4
79 000000A4 7E14                move #20,d7             * CLRSCREEN
80 000000A6 4EAC0420            jsr $420(a4)
81                              *
82                              * Ende Grundprogrammbearbeitung
83                              ********************************************************
84                              * dann zurueckspeichern der Datei
85                              *
86 000000AA 41F900009000        lea ladeadr,a0          * Suchen des Endes
87                              lpa:
88 000000B0 0C180000            cmp.b #0,(a0)+          * =0
89 000000B4 66FA                bne lpa
90 000000B6 117C001AFFFF        move.b #$1a,-1(a0)      * Endekennung
91 000000BC 223900000000        move.l fcbadr,d1
92 000000C2 2079000000000       movea.l fcbadr,a0       * fuer Current Rec...
93 000000C8 4228000C            clr.b 12(a0)            * EX=0
94 000000CC 42280020            clr.b 32(a0)            * CR=0
95 000000D0 700F                move #open,d0
96 000000D2 4E42                trap #2                 * OPEN
97 000000D4 B07C00FF            cmp #255,d0
98 000000D8 6748                beq error               * Fehler System
99 000000DA 223C00009000        move.l #ladeadr,d1      * Adresse Quelle
100                             schleif1:
101 000000E0 48E74000           movem.l d1,-(a7)
102 000000E4 701A               move #setdma,d0
103 000000E6 4E42               trap #2
104 000000E8 223900000000       move.l fcbadr,d1
105 000000EE 7015               move #write,d0
106 000000F0 4E42               trap #2                 * und schreiben
107 000000F2 4CDF0002           movem.l (a7)+,d1
108 000000F6 4A40               tst d0
109 000000F8 6628               bne error               * Fehler aufgetreten
110 000000FA 2041               movea.l d1,a0           * suchen nach 1A
111 000000FC 363C007F           move #128-1,d3
112                             sulp:
113 00000100 0C18001A           cmp.b #$1a,(a0)+
114 00000104 670C               beq finlp
115 00000106 51CBFFF8           dbra d3,sulp
116 0000010A D2BC00000080       add.l #128,d1           * naechste Adresse
117 00000110 60CE               bra schleif1            * bis 1A aufgetaucht.
118                             finlp:
119 00000112 4228FFFF           clr.b -1(a0)            * und wieder loeschen.
120 00000116 223900000000       move.l fcbadr,d1
121 0000011C 7010               move #close,d0
122 0000011E 4E42               trap #2
123 00000120 600A               bra exit                * ok Ende
124
125                             error:
126 00000122 223C00000132       move.l #errtxt,d1
127 00000128 7009               move #printstr,d0
128 0000012A 4E42               trap #2
129                             exit:
130 0000012C 7000               move #reboot,d0
131 0000012E 4E42               trap #2
132 00000130 4E75               rts
133
134
135                             errtxt:
136 00000132 0D0A4665686C6572   dc.b 13,10,'Fehler beim anlegen'
136 0000013A 206265696D20616E
136 00000142 6C6567656E
137 00000147 24                 dc.b '$'
138
139 00000000                    .data
140                             .even
141
142 00000000 00000000           fcbadr: dc.l 0
143 00000004 00000000           gruadr: dc.l 0
144
145
146
147
148
149
150
151
152
153
154 00000008                    end
```

```
C P / M   6 8 0 0 0   A s s e m b l e r                    Revision 04.03
Source File: assrdk.s

  1                     ************************************************
  2                     * Assembler   mit Grundprogrammunterstuetzung *
  3                     * (C) 1984 Rolf-Dieter Klein   841021  V1.0   *
  4                     ************************************************
  5
  6                     * Grundprogrammaufruf mit Hilfe Funktion 23
  7                     * im BIOS
  8                     * Ladeadresse ist $9000
  9
 10                     ladeadr equ     $9000
 11
 12                     reboot  equ     0
 13                     open    equ     15
 14                     setdma  equ     26
 15                     make    equ     22
 16                     read    equ     20
 17                     write   equ     21
 18                     close   equ     16
 19                     printstr equ    9
 20
 21 00000000             .text
 22
 23
 24                     start:
 25 00000000 4E560000    link a6,#0              * mark stack frame
 26 00000004 206E0008    move.l 8(a6),a0         * Adresse BASIS-Seite
 27 00000008 43E8005C    lea $5c(a0),a1          * FCB
 28 0000000C 2209        move.l a1,d1            * Adresse FCB laden
 29 0000000E 23C100000000 move.l d1,fcbadr       * dort merken
 30 00000014 303C000F    move.w #open,d0
 31 00000018 4E42        trap #2                 * OPEN
 32 0000001A B07C00FF    cmp #255,d0             * =255 dann Datei nicht da
 33 0000001E 670000BC    beq error               * dann Fehler
 34                     okda:                   * Datei ist schon da
 35                                             * erst mal einlesen
 36 00000022 223C00009000 move.l #ladeadr,d1     * Startadresse
 37                     schleife:
 38 00000028 48E74000    movem.l d1,-(a7)
 39 0000002C 701A        move #setdma,d0
 40 0000002E 4E42        trap #2                 * DMA setzen
 41 00000030 223900000000 move.l fcbadr,d1
 42 00000036 7014        move #read,d0
 43 00000038 4E42        trap #2                 * LESEN
 44 0000003A 4CDF0002    movem.l (a7)+,d1
 45 0000003E D2BC000000B0 add.l #128,d1          * neue Adresse
 46 00000044 4A40        tst d0
 47 00000046 67E0        beq schleife            * ok eingelesen
 48 00000048 41F900009000 lea ladeadr,a0         * Suchen Nach 1A und ersetzen durch
 49                     lpp:
 50 0000004E 0C1B001A    cmp.b #$1a,(a0)+
 51 00000052 66FA        bne lpp
 52 00000054 4228FFFF    clr.b -1(a0)            * loeschen vorhergehendes Bytes
 53 00000058 23C800000008 move.l a0,lastbyte
 54                     eingelesen:             * Schliessen, Aufruf des Editors
 55 0000005E 223900000000 move.l fcbadr,d1
 56 00000064 7010        move #close,d0
 57 00000066 4E42        trap #2                 * Datei schliessen
 58                     ************************************************
 59                     * Assembler Grundprogramm aufrufen
 60                     * einfache Version, laedt Quelle erst ganz in Speicher
 61                     * dann wird uebersetzt, Ergebnis bleibt im Speicher
 62                     *
 63 00000068 7017        move #23,d0
 64 0000006A 4E43        trap #3                 * Sonderfunktion
 65 0000006C 23CC00000004 move.l a4,gruadr
 66                     *
 67 00000072 203C00009000 move.l #ladeadr,d0
 68 00000078 7E43        move #67,d7
 69 0000007A 4EAC0420    jsr $420(a4)            * PUTSTX
 70 0000007E 287900000004 movea.l gruadr,a4
 71 00000084 7E34        move #52,d7
 72 00000086 4EAC0420    jsr $420(a4)            * NIL
 73 0000008A 287900000004 movea.l gruadr,a4
 74 00000090 7E44        move #68,d7
 75 00000092 4EAC0420    jsr $420(a4)            * GETORG
 76 00000096 2F00        move.l d0,-(a7)         * alte Org-Adresse
 77 00000098 287900000004 movea.l gruadr,a4
 78 0000009E 203900000008 move.l lastbyte,d0     * letztes Byte, bei Debug Platz dor
krit.
 79 000000A4 D0BC000000C8 add.l #200,d0          * Sicherheitsfaktor
 80 000000AA 08B000000   bclr #0,d0              * EVEN-Adresse
 81 000000AE 7E45        move #69,d7
 82 000000B0 4EAC0420    jsr $420(a4)            * PUTORG neue Adresse
 83 000000B4 287900000004 movea.l gruadr,a4
 84 000000BA 7E41        move #65,d7
 85 000000BC 4EAC0420    jsr $420(a4)            * ASSEMBLER
 86 000000C0 201F        move.l (a7)+,d0         * alte Org-Adresse
 87 000000C2 287900000004 movea.l gruadr,a4
 88 000000C8 7E45        move #69,d7
 89 000000CA 4EAC0420    jsr $420(a4)            * PUTORG alte Adresse
 90 000000CE 287900000004 movea.l gruadr,a4
 91 000000D4 7E31        move #49,d7
 92 000000D6 4EAC0420    jsr $420(a4)            * CRT
 93                     *
 94                     * Ende Grundprogrammbearbeitung
 95                     ********************************************************
 96 000000DA 600A        bra exit                * ok Ende
 97
 98                     error:
 99 000000DC 223C000000EC move.l #errtxt,d1
100 000000E2 7009        move #printstr,d0
101 000000E4 4E42        trap #2
102                    exit:
103 000000E6 7000        move #reboot,d0
104 000000E8 4E42        trap #2
105 000000EA 4E75        rts
106
107                    errtxt:
108 000000EC 0D0A4665686C6572  dc.b 13,10,'Fehler, Datei nicht da'
108 000000F4 2C20446174656920
108 000000FC 6E69636874206461
109 00000104 24             dc.b '$'
110
111 00000106                .data
111 00000000
112                         .even
113
114 00000000 00000000      fcbadr: dc.l 0
115 00000004 00000000      gruadr: dc.l 0
116 00000008 00000000      lastbyte: dc.l 0
117
118 0000000C                end
```

Bild 20. Den Assembler im Grundprogramm unter CP/M ansprechen

Adresse $8000 muß dabei RAM sein, was bei CP/M ja der Fall ist. CP/M-68k wird bei 8 Zoll auf insgesamt 5 Disketten geliefert, im NDR80-Format sind es weniger. Das ECMA70-Format wird von uns nicht mehr verwendet, da der Platz zu gering ist, um sinnvoll damit zu arbeiten. In CP/M-68k ist nicht nur ein Assembler, sondern auch ein Compiler für die Sprache „C" eingearbeitet, der auch Gleitkommaarithmetik beherrscht.

„C" ist eine Programmiersprache, die in der Systemprogrammierung sehr verbreitet ist, da man in ihr sehr kompakte und schnelle Programme schreiben kann.

Hilfsprogramme unter CP/M-68k

Auf den Systemdisketten befindet sich auch ein Editor, der jedoch nur zeilenorientiert ist. Da wir im Grundprogramm schon über einen bildschirmorientierten Editor verfügen, liegt der Gedanke nahe, beides zu kombinieren.
Bild 19 zeigt das fertige Programm. Das Programm wird mit dem CP/M-68k-Assembler übersetzt und als EDITRDK.68K-Datei gebunden. Dann kann man einfach Programme editieren, wenn man unter CP/M eingibt:
EDITRDK name.ext
Der Text wird dann automatisch geladen, falls die Datei schon existierte, und dann der Editor im Grundprogramm aufgerufen. Wenn man den Editorvorgang mit CTRL-K und X beendet, so wird das Programm wieder auf Diskette zurückgeschrieben und man landet wieder im CP/M-Betriebssystem.

```
C P / M   6 8 0 0 0   A s s e m b l e r                    Revision 04.03
Source File: starte.s

   1                          ************************************************
   2                          * Starteprog. mit Grundprogrammunterstuetzung *
   3                          * (C) 1984 Rolf-Dieter Klein   841021  V1.0   *
   4                          ************************************************
   5
   6                          * Starte   name   , Grundprogrammname wird ausgefuehrt.
   7                          * Grundprogrammaufruf mit Hilfe Funktion 23
   8                          * im BIOS
   9                          * Ladeadresse ist $9000
  10
  11                          ladeadr equ     $9000
  12
  13                          reboot  equ     0
  14                          open    equ     15
  15                          setdma  equ     26
  16                          make    equ     22
  17                          read    equ     20
  18                          write   equ     21
  19                          close   equ     16
  20                          printstr equ    9
  21                          conout  equ     2
  22
  23 00000000                  .text
  24
  25                          start:
  26 00000000 4E560000         link a6,#0              * mark stack frame
  27 00000004 206E0008         move.l 8(a6),a0         * Adresse BASIS-Seite
  28 00000008 43E80080         lea $80(a0),a1          * DMA-Adr, Kommando nach Befehl
  29 0000000C 2F09             move.l a1,-(a7)
  30 0000000E 223C0000003C     move.l #starttxt,d1
  31 00000014 7009             move #printstr,d0
  32 00000016 4E42             trap #2
  33 00000018 225F             movea.l (a7)+,a1
  34 0000001A 5289             addq.l #1,a1            * Auf erstes Byte
  35                          loop:
  36 0000001C 1219             move.b (a1)+,d1
  37 0000001E 6706             beq floop
  38 00000020 7002             move #conout,d0
  39 00000022 4E42             trap #2
  40 00000024 60F6             bra loop
  41                          floop:
  42 00000026 103C000D         move.b #$d,d0
  43 0000002A 7002             move #conout,d0
  44 0000002C 4E42             trap #2
  45 0000002E 103C000A         move.b #$a,d0
  46 00000032 7002             move #conout,d0
  47 00000034 4E42             trap #2
  48                          *
  49 00000036 7000             move #reboot,d0
  50 00000038 4E42             trap #2
  51 0000003A 4E75             rts
  52
  53
  54 0000003C                  .text
  55                          .even
  56
  57                              starttxt:
  58 0000003C 0D0A537461727420      dc.b $d,$a,'Start des Programms',$d,$a
  58 00000044 6465732050726F67
  58 0000004C 72616D6D730D0A
  59 00000053 01452024              dc.b 1,'E $'
  60
  61
  62 00000057                       end
```

Bild 21. Starten von Programmen

Alle Grundprogrammfunktionen können mit diesem Mechanismus im BIOS aufgerufen werden. Dazu enthält unser BIOS einen weiteren TRAP-Eingang mit dem Index 23. Damit ist es möglich, die Startadresse des Grundprogramms festzustellen und dort steht ab Adresse Start +$400 eine Sprungtabelle auf die einzelnen Funktionen. Den TRAP-Aufruf TRAP #1 kann man normalerweise unter CP/M nicht verwenden, denn er wird vom Betriebssystem überschrieben. Man könnte ihn aber selbst wieder zurücksetzen und dann wie gewohnt arbeiten. Ein weiteres nützliches Programm ist in *Bild 20* gezeigt. Damit kann man Programme von der Diskette weg in Assembler übersetzen, und zwar mit dem eingebauten Assembler des Grundprogramms. Das geht wesentlich schneller als mit dem CP/M-68k-Assembler, der sehr lang ist (50 KByte). *Bild 21* zeigt ein Programm, mit dem man das übersetzte Programm dann starten kann, wenn man den Namen angibt, der als Symbol definiert wurde. *Bild 22* zeigt ein Programm, das einen Text in den Speicher lädt und zwar ab Adresse $9000. Der Text muß zuvor auf der Diskette stehen. Und mit dem Programm in *Bild 23* schließlich kann man einen Text, der ab Adresse $9000 steht, auf Diskette abspeichern.
Achtung. Wenn man das Grundprogramm aufruft, so wird unter CP/M der Textstart nach dem Einschalten bei $98000 oder $E9000 liegen, man muß also den Textstart ändern, wenn man an den geladenen Text heran will. Doch man kann natürlich auch einfach anhand des Listings die Programme seinen eigenen Bedürfnissen anpassen. Das Arbeiten unter CP/M-68k ist manchmal nicht ganz einfach. Übung macht den Meister und wir wünschen allen viel Freude und Spaß beim Programmieren.

```
C P / M   6 8 0 0 0   A s s e m b l e r                    Revision 04.03
Source File: lade.s

                                  ***********************************************
                                  * Ladeprogramm fuer Quellen                   *
                                  * (C) 1984 Rolf-Dieter Klein   841021  V1.0   *
                                  ***********************************************

                                  * Grundprogrammaufruf mit Hilfe Funktion 23
                                  * im BIOS
                                  * Ladeadresse ist $9000

                                  ladeadr equ     $9000

                                  reboot  equ     0
                                  open    equ     15
                                  setdma  equ     26
                                  make    equ     22
                                  read    equ     20
                                  write   equ     21
                                  close   equ     16
                                  printstr equ    9

00000000                           .text


                                  start:
00000000 4E560000                  link a6,#0              * mark stack frame
00000004 206E0008                  move.l 8(a6),a0         * Adresse BASIS-Seite
00000008 43E8005C                  lea $5c(a0),a1          * FCB
0000000C 2209                      move.l a1,d1            * Adresse FCB laden
0000000E 23C100000000              move.l d1,fcbadr        * dort merken
00000014 303C000F                  move.w #open,d0
00000018 4E42                      trap #2                 * OPEN
0000001A B07C00FF                  cmp #255,d0             * =255 dann Datei nicht da
0000001E 6742                      beq error               * dann Fehler
                                  okda:                    * Datei ist schon da
                                                           * erst mal einlesen
00000020 223C00009000              move.l #ladeadr,d1      * Startadresse
                                  schleife:
00000026 48E74000                  movem.l d1,-(a7)
0000002A 701A                      move #setdma,d0
0000002C 4E42                      trap #2                 * DMA setzen
0000002E 223900000000              move.l fcbadr,d1
00000034 7014                      move #read,d0
00000036 4E42                      trap #2                 * LESEN
00000038 4CDF0002                  movem.l (a7)+,d1
0000003C D2BC00000080              add.l #128,d1           * neue Adresse
00000042 4A40                      tst d0
00000044 67E0                      beq schleife            * ok eingelesen
00000046 41F900009000              lea ladeadr,a0          * Suchen Nach 1A und ersetzen durch
                                  lpp:
0000004C 0C18001A                  cmp.b #$1a,(a0)+
00000050 66FA                      bne lpp
00000052 4228FFFF                  clr.b -1(a0)            * loeschen vorhergehendes Bytes
                                  eingelesen:              * Schliessen, Aufruf des Editors
00000056 223900000000              move.l fcbadr,d1
0000005C 7010                      move #close,d0
0000005E 4E42                      trap #2                 * Datei schliessen
00000060 600A                      bra exit                * ok Ende

                                  error:
00000062 223C00000072              move.l #errtxt,d1
00000068 7009                      move #printstr,d0
0000006A 4E42                      trap #2
                                  exit:
0000006C 7000                      move #reboot,d0
0000006E 4E42                      trap #2
00000070 4E75                      rts

                                  errtxt:
00000072 0D0A4665686C6572          dc.b 13,10,'Fehler Datei nicht da'
0000007A 204461746569206E
00000082 69636874206461
00000089 24                        dc.b '$'

00000000                           .data
                                   .even

00000000 00000000                 fcbadr: dc.l 0
00000004 00000000                 gruadr: dc.l 0

00000008                           end
```

Bild 22. Texte in den Speicher laden

```
C P / M   6 8 0 0 0   A s s e m b l e r                    Revision 04.03
Source File: save.s

                          ***********************************************
                          * SAVE-Programm, legt Textgebiet auf Diskette *
                          * (C) 1984 Rolf-Dieter Klein   841021  V1.0   *
                          ***********************************************

                          * Grundprogrammaufruf mit Hilfe Funktion 23
                          * im BIOS
                          * Ladeadresse ist $9000


                          ladeadr equ     $9000

                          reboot  equ     0
                          open    equ     15
                          setdma  equ     26
                          make    equ     22
                          read    equ     20
                          write   equ     21
                          close   equ     16
                          printstr equ    9

00000000                   .text


                          start:
00000000 4E560000          link a6,#0              * mark stack frame
00000004 206E0008          move.l 8(a6),a0         * Adresse BASIS-Seite
00000008 43E8005C          lea $5c(a0),a1          * FCB
0000000C 2209              move.l a1,d1            * Adresse FCB laden
0000000E 23C100000000      move.l d1,fcbadr        * dort merken
00000014 303C000F          move.w #open,d0
00000018 4E42              trap #2                 * OPEN
0000001A B07C00FF          cmp #255,d0             * =255 dann Datei nicht da
0000001E 6612              bne okda                * dann kreieren
00000020 223900000000      move.l fcbadr,d1
00000026 303C0016          move.w #make,d0
0000002A 4E42              trap #2                 * anlegen
0000002C 0C4000FF          cmpi #255,d0
00000030 6760              beq error               *
                          okda:                    * dann zurueckspeichern
                          *
00000032 41F900009000      lea ladeadr,a0          * Suchen des Endes
                          lpa:
00000038 0C10001A          cmp.b #$1a,(a0)         * auch ok
0000003C 670C              beq lpb
0000003E 0C180000          cmp.b #0,(a0)+          * =0
00000042 66F4              bne lpa
00000044 117C001AFFFF      move.b #$1a,-1(a0)      * Endekennung, falls noch nicht da
                          lpb:
0000004A 223C00009000      move.l #ladeadr,d1      * Adresse Quelle
                          schleif1:
00000050 48E74000          movem.l d1,-(a7)
00000054 701A              move #setdma,d0
00000056 4E42              trap #2
00000058 223900000000      move.l fcbadr,d1
0000005E 7015              move #write,d0
00000060 4E42              trap #2                 * und schreiben
00000062 4CDF0002          movem.l (a7)+,d1
00000066 4A40              tst d0
00000068 6628              bne error               * Fehler aufgetreten
0000006A 2041              movea.l d1,a0           * suchen nach 1A
0000006C 363C007F          move #128-1,d3
                          sulp:
00000070 0C18001A          cmp.b #$1a,(a0)+
00000074 670C              beq finlp
00000076 51CBFFF8          dbra d3,sulp
0000007A D2BC00000080      add.l #128,d1           * naechste Adresse
00000080 60CE              bra schleif1            * bis 1A aufgetaucht.
                          finlp:
00000082 4228FFFF          clr.b -1(a0)            * wieder auf 0
00000086 223900000000      move.l fcbadr,d1
0000008C 7010              move #close,d0
0000008E 4E42              trap #2
00000090 600A              bra exit                * ok Ende

                          error:
00000092 223C000000A2      move.l #errtxt,d1
00000098 7009              move #printstr,d0
0000009A 4E42              trap #2
                          exit:
0000009C 7000              move #reboot,d0
0000009E 4E42              trap #2
000000A0 4E75              rts


                          errtxt:
000000A2 0D0A4665686C6572  dc.b 13,10,'Fehler beim anlegen'
000000AA 206265696D20616E
000000B2 6C6567656E
000000B7 24                dc.b '$'

00000000                   .data
                           .even

00000000 00000000         fcbadr: dc.l 0
00000004 00000000         gruadr: dc.l 0

00000008                   end
```

Bild 23. Damit kann man Texte auf der CP/M-68k-Diskette abspeichern

Dr. Hans Hehl

Der Grafikprozessor EF9366

Beim NDR-Klein-Computer wird eine sogenannte GDP64K-Baugruppe verwendet. Ihr Aufbau und der Funktionstest wurden im Sonderheft Nr. 88, „Mikrocomputer Schritt für Schritt, Teil 1", beschrieben. Nun soll der dabei verwendete Grafikprozessor EF9366 der Firma Thomson etwas genauer untersucht werden.

Die drei Buchstaben G, D, P sind die Anfangsbuchstaben der drei Wörter „GRAPHIC DISPLAY PROCESSOR". Die Zahl 64 gibt die Speicherkapazität (64 KByte) des Bildwiederholspeichers an.

Im Prinzip ist die Karte ein eigenständiger Mikrocomputer mit eigenem Hauptspeicher. Vom GDP werden vor allem die zur Ansteuerung eines Monitors notwendigen Signale erzeugt, also das Vertikal- und das Horizontal-Synchronsignal, sowie das Videosignal, die Bildinformation. Diese wird aus dem Hauptspeicher (Bildwiederholspeicher) entnommen, wobei einem dunklen Bildpunkt ein gesetztes Bit entspricht. Genauere Einzelheiten über die Erzeugung eines Monitorbildes sind im Buch „Mikrocomputer selbstgebaut und programmiert" von R.-D. Klein geschildert (auch das Studium des Datenblattes des Prozessors ist letztlich zu empfehlen).

Registeradresse	A3	A2	A1	A0	Registerfunktion
70h	0	0	0	0	Status-Register, nur lesend (R/W = 1) Befehlsregister (CMD), nur schreibend (R/W = 0)
71h	0	0	0	1	1. Kontrollregister, Schreibvorgänge u. Interruptsteuerung
72h	0	0	1	0	2. Kontrollregister, Zeichenstrichart der Vektoren, Orientierung der Buchstaben
73h	0	0	1	1	Register für Zeichengröße in X- u. Y-Achse (C-SIZE)
74h	0	1	0	0	nicht verwendet
75h	0	1	0	1	Delta-X-Register (siehe Vektorbefehle)
76h	0	1	1	0	nicht verwendet
77h	0	1	1	1	Delta-Y-Register (siehe Vektorbefehle)
78h	1	0	0	0	X-Register, Bit 0 - 3 (MSB) X-Koordinaten (256 - 4095)
79h	1	0	0	1	X-Register, Bit 0 - 7 (LSB) X-Koordinaten (0 - 255)
7Ah	1	0	1	0	Y-Register, Bit 0 - 3 (MSB) Y-Koordinate (256 - 4095)
7Bh	1	0	1	1	Y-Register, Bit 0 - 7 (LSB) Y-Koordinate (0 - 255)
7Ch	1	1	0	0	X-Lichtgriffelregister, nur lesend Bit 2-7, Bit 1 immer 0, Bit 0 = Status
7Dh	1	1	0	1	Y-Lichtgriffelregister, nur lesend Bit 0 - 7
7Eh	1	1	1	0	nicht verwendet
7Fh	1	1	1	1	nicht verwendet

Bild 2. Die Register des Prozessors 9366

Was der EF9366 kann

Vektoren werden mit bis zu 1,5 Millionen Bildpunkten pro Sekunde ausgegeben. Ein integrierter Zeichengenerator erzeugt in einer 5×8-Bildpunkte-Matrix die Zeichen des amerikanischen Stan-

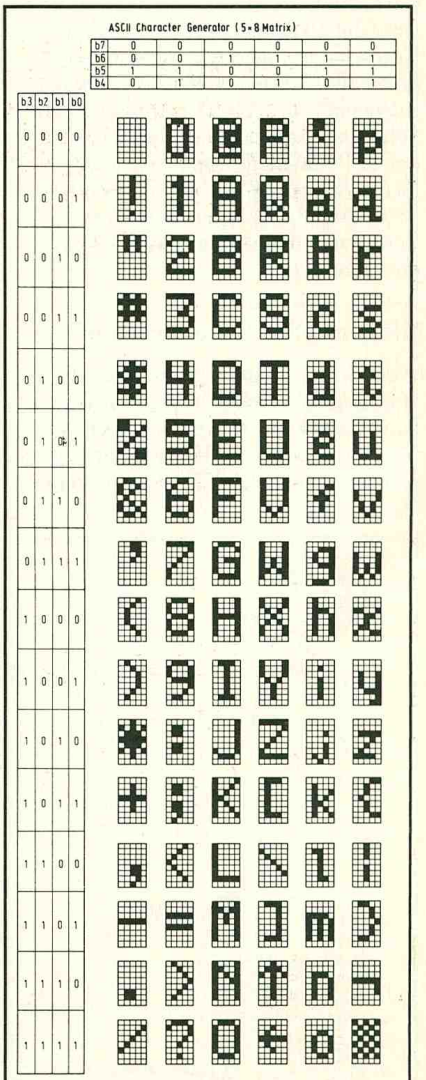


Bild 1. Der ASCII-Zeichensatz, der in dem EF-9366 eingebaut ist

dard-ASCII-Zeichensatzes aus den binär codierten Zahlen von 20 bis 127 (*Bild 1*). Leider gibt es keine Umlaute und kein „ß". Die vorhandenen Zeichen sind in fast beliebiger Form auf dem Bildschirm darstellbar.

Alle Register ziehen?

Der Prozessor erhält die dafür notwendigen Informationen über verschiedene Ein- und Ausgabeeinheiten, sogenannte Register. In diese Register können Zahlenwerte von einem Hauptrechner eingeschrieben werden, die dann dort gespeichert und ausgewertet werden. Umgekehrt kann der Prozessor über diese Register Informationen ausgeben.
Der GDP wird beim NDR-Klein-Computer über die Adressenleitungen A7...A4 mit dem sedezimalen Wert 7 (binär 0111, Bit 7...4) angesprochen. Die Auswahl der 16 Register erfolgt mit den Adressenleitungen A3...A0 mit den sedezimalen Werten 0...F, also mit den Adressen 70h...7Fh. Der Adressenwert 70h ist am Adressendecoder 74LS138 der GDP64K-Baugruppe fest eingestellt, da der Ausgang Y7 (Pin 7) verwendet wird. Von diesen 16 Registern werden nur 12 verwendet, wie aus *Bild 2* zu entnehmen ist.

```
gdp:=  70            ;GDP-Registeradresse
se:=  60             ;Seitenportadresse
ci:=  24             ;für SBC2: Zeicheneingabe

8800    CD 882A      call warte      ;warten bis bereit
8803    AF           xor  a          ;Akku löschen
8804    D3 60        out  (se),a     ;Seite 0 anwählen
8806    CD 882A      call warte      ;warten bis fertig
8809    3E 03        ld   a,3h       ;3h = 0011
                                     ;Bit 0 gesetzt: Stift unten
                                     ;Bit 1 gesetzt: schreiben
880B    D3 71        out  (gdp+1),a  ;ausgeben an GDP
880D    CD 882A      call warte      ;warten bis fertig

                     aus:
8810    CD 0024      call ci         ;Zeichen von Tastatur
8813    FE 1B        cp   1bh        ;Escape-Taste ?
8815    C8           ret  z          ;dann zurück zum Menue
8816    F5           push af         ;Akku-Wert retten
8817    CD 8822      call era        ;jetzt altes Zeichen löschen
881A    CD 882A      call warte      ;warten bis fertig
881D    F1           pop  af         ;alten Akku-Wert regenieren
881E    D3 70        out (gdp),a     ;neues Zeichen an GDP
8820    18 EE        jr   aus        ;nochmal das Ganze

                     era:
8822    CD 882A      call warte      ;warten bis bereit
8825    3E 06        ld   a,6h       ;6h = 0110b, Bit 1 u. 2 gesetzt
                                     ;X-, Y-Register u. Bildschirm
                                     ;löschen
8827    D3 70        out  (gdp),a    ;Befehl an CMD-Register
8829    C9           ret             ;zurück

                     warte:
882A    DB 70        in   a,(gdp)    ;Status abfragen, ob fertig
882C    E6 04        and  4h         ;4h = 0100b, Maske Bit 2
882E    28 FA        jr   z,warte    ;Bit 2 = 0, nochmal warten
8830    C9           ret             ;zurück
```

Bild 4. Dieses Z80-Programm gibt ebenfalls ASCII-Zeichen aus

Nicht nur das Grundprogramm

Welche Befehle der GDP zur Ausgabe von ASCII-Zeichen benötigt, kann anhand eines kleinen Programmes ausprobiert werden. Nach Drücken einer Taste soll das jeweilige ASCII-Zeichen in der linken unteren Bildschirmecke erscheinen.

Wenn Sie diese Aufgabe mit einem Programm lösen wollen, dann können Sie einige Befehle des Grundprogrammes verwenden, wie in *Bild 3* gezeigt. *Bild 4* zeigt ein Programm, bei dem der GDP ohne Befehle aus dem Grundprogramm angesteuert wird. Dabei ist zu beachten, daß nur der Befehl CD 0024 (Adresse 8810) als CD 0024.W einzugeben ist. Auch unter CP/M laufen die Beispielprogramme, wenn kleine Anpassungen vorgenommen werden. Das Programm muß hier bei der Adresse 100h beginnen. *Bild 5* enthält hierzu das Listing des gleichen Programmes als Assembleraus-druck. Ebenso geht es mit dem 8-KByte-Basic (in zwei EPROMs auf SBC2), wobei der Ausgabebefehl OUT (Portadresse, Wert) und der Befehl WAIT (Portadresse, Y, Z) für die Abfrage des Statusregisters des GDP Verwendung finden. *Bild 6* zeigt das dazugehörige Basic-Programm. Diese Beispielprogramme sind möglichst einfach gehalten und sollen zum Experimentieren anregen.

```
GDP:= 70
SEITE:= 60
Taste:= 8810

8800    CD WAIT          warten bis GDP bereit
8803    AF               Akku löschen
8804    D3 SEITE         Seite 0 anwählen
8806    CD WAIT          warten bis GDP bereit
8809    3E 03            Akku mit 3 laden
                         3h = 0011
                         Bit 0 gesetzt: Stift unten
                         Bit 1 gesetzt: schreiben
880B    D3 GDP+1         ausgeben an GDP
880D    CD WAIT          warten bis GDP fertig

8810    CD CI            Zeichen von Tastatur
8813    FE 1B            Escape-Taste gedrückt?
8815    C8               dann zurück zum Menue
8816    F5               Zeichen auf Stapel retten
8817    CD WAIT          warten bis GDP fertig
881A    3E 06            Akku mit 6 laden (löschen; X,Y = 0)
881C    D3 GDP           ausgeben an GDP
881E    CD WAIT          warten bis GDP fertig
8821    F1               Zeichen wieder holen
8822    D3 GDP           an GDP ausgeben
8824    C3 TASTE         zurück zu 8810
```

Bild 3. So können unter dem Z80-Grundprogramm ASCII-Zeichen ausgegeben werden

Neue Bekanntschaft

Anhand des Z80-Maschinenprogrammes für die SBC2 (Bild 4) fällt es nicht schwer, den GDP kennenzulernen. Die einzelnen Programmschritte lassen sich

mit dem Grundprogramm-Menü <3 = Einzelschritt> verfolgen.
Am Programmbeginn empfiehlt es sich, das Statusport 70h abzufragen, ob der GDP bereit ist. Dies geschieht in einem Unterprogramm „warte" mit einer Abfrageschleife, bei der mit der Bitmaske 0100 durch eine UND-Verknüpfung mit dem Akku-Wert der Z80-CPU so lange gewartet wird, bis Bit 2 des Statusports den Wert 1 annimmt. Dann erst kann der GDP neue Befehle entgegennehmen. Mit dem Befehl JR Z,WARTE erfolgt der Rücksprung zum Unterprogrammanfang.
Zu Beginn wird durch Beschreiben des Seitenports (Adresse 60h) mit dem Wert 0 die Schreib- und Leseseite 0 eingestellt. Insgesamt gibt es vier Bildseiten zu je 512 × 256 Bildpunkten. Ausgewählt wird die Schreibseite mit Bit 7 und 6 und die Leseseite mit Bit 5 und 4 des an dem Seitenport ausgegebenen sedezimalen Wertes. Bei zwei Bits sind vier Möglichkeiten vorhanden, nämlich die Bitkombinationen 00, 01, 10 und 11, die den Bildseiten 0, 1, 2 und 3 entsprechen. Es kann eine Bildseite beschrieben und gleichzeitig eine andere angezeigt werden. Der sedezimale Wert 70 (01110000) ergibt die Schreibseite Nr. 1 und die Leseseite Nr. 3. Zur gleichzeitigen Darstellung zweier Bildseiten muß schnell zwischen diesen hin- und hergeschaltet werden.
Ab Programm-Adresse 8809 wird der Schreibmodus im ersten Kontrollregister (Adresse 71h) festgelegt. Verwendung finden die Bits 0...6 des Registers. Ihre Bedeutung ist in *Bild 7* enthalten. Da wir schreiben wollen, müssen wir Bit 0 und 1 auf den Wert 1 setzen. Dies geschieht durch Einschreiben des Wertes 3 (binär 0011) in das Register. Wenn wir statt dessen den sedezimalen Wert B (binär 1011) in das Register schreiben, kann die Bildseite nicht über den Rand hinaus (dann unsichtbar) beschrieben werden und es wird auf der gleichen Bildzeile von vorne begonnen.
Nach einem Wartezyklus wird mittels Grundprogrammbefehl CD CI ein Zeichen von der Tastatur in den Akku geladen. Die Abfrage nach dem Escape-Zeichen (1Bh) ermöglicht einen Ausstieg ins Menü. Nun wird ein schon ausgegebenes Zeichen mit dem Unterprogramm „ERA" gelöscht. Hier gibt es verschiedene Möglichkeiten, die sich durch Austauschen des Werts an der Adresse 8826 ausprobieren lassen. *Bild 8* zeigt die Bit-

```
 10 CLRS
 20 GDP = HEX("70")
 30 INPUT"eine Taste u. CR druecken";A$
 35 CLRS
 40 IF ASC(A$) = 27 THEN 100          :REM Escape beendet
 50 PAGE 0,0                          :REM Seite 0
 55 MOVETO 0,0                        :REM linke, untere Ecke
 60 OUT GDP, ASC(A$)                  :REM 1.Zeichen an GDP
 70 WAIT GDP,4,0                      :REM warten bis Bit 2 = 1
 80 POKE HEX("87C5"),0                :REM Cursor aus
 90 GOTO 30
100 END
```

Bild 6. Das Programm in Basic für die Platine SBC2

```
Bit 0       1 : schreiben          0 : nicht schreiben

Bit 1       1 : nicht löschen      0 : löschen

Bit 2       1 : Schnell schreiben ohne Bildschirmausgabe

Bit 3       1 : nur im Bildfenster schreiben
            0 : Bildfenster = 4096 * 4096 Bildpunkte

Bit 4       1 : Interrupt-Freigabe nach einer Lichtgriffel-
                sequenz

Bit 5       1 : Interrupt-Freigabe bei jedem Vertikal-
                Synchronsignal

Bit 6       1 : Interrupt-Freigabe, wenn der GDP bereit

Bit 7       0   nicht verwendet
```

Bild 7. Die Bedeutung der Bits im Kontrollregister. Bit 0 und 1 werden auch durch das Befehlsregister verändert

```
gdp   equ   70h         ;GDP-Registeradresse
se    equ   60h         ;Seitenportadresse
ci    equ   0F003H      ;MC-Comp. Zeicheneingabe

0100    CD 012C     call warte      ;warten bis bereit
0103    AF          xor  a          ;Akku löschen
0104    D3 60       out  (se),a     ;Seite 0 anwählen
0106    CD 012C     call warte      ;warten bis fertig
0109    3E 03       ld   a,3h       ;3h = 0011
                                      ;Bit 0 gesetzt: Stift unten
                                      ;Bit 1 gesetzt: schreiben
010B    D3 71       out  (gdp+1),a  ;ausgeben an GDP
010D    CD 012C     call warte      ;warten bis fertig

0110                aus:
0110    CD F003     call ci         ;Zeichen von Tastatur
0113    FE 1B       cp   1bh        ;Escape-Taste ?
0115    CA 0000     jp   z,0        ;dann zurück zu CP/M
0118    F5          push af         ;Akku-Wert retten
0119    CD 0124     call era        ;jetzt altes Zeichen löschen
011C    CD 012C     call warte      ;warten bis fertig
011F    F1          pop  af         ;alten Akku-Wert regenieren
0120    D3 70       out  (gdp),a    ;neues Zeichen an GDP ausgeben
0122    18 EC       jr   aus        ;nochmal das Ganze

0124                era:
0124    CD 012C     call warte      ;warten bis fertig
0127    3E 06       ld   a,6h       ;6h = 0110b, Bit 1 u. 2 gesetzt
                                    ;X-, Y-Register u. Bildschirm
                                    ;löschen
0129    D3 70       out  (gdp),a    ;Befehl an CMD-Register
012B    C9          ret             ;zurück

012C                warte:
012C    DB 70       in   a,(gdp)    ;Status abfragen, ob fertig
012E    E6 04       and  4h         ;4h = 0100b, Maske Bit 2
0130    28 FA       jr   z,warte    ;Bit 2 = 0, nochmal warten
0132    C9          ret             ;zurück
```

Bild 5. Dieses Programm gibt die ASCII-Zeichen unter CP/M aus

belegung des CMD-Registers (Adresse 70h). Der binäre Wert 0100 (4) löscht nur den Bildschirm, ohne die Koordinaten der nächsten Schreibposition zu löschen. Der Wert 0101 (5) setzt die Schreibkoordinaten X, Y auf 0, ohne den Bildschirm zu löschen. Schreiben wir den binären Wert 0110 (6) in das Register, so wird der Bildschirm gelöscht und mit dem Schreiben an dem Koordinatenpunkt 0,0 (linke untere Ecke) begonnen. Natürlich muß der Wert des eingegebenen Zeichens mit der Befehlssequenz PUSH AF und POP AF erhalten werden.
Ab Adresse 881E wird dann das Zeichen ausgegeben und das Ganze beginnt bei Adresse 8810 von vorne.

Zeichenzauber oder besondere Wünsche

Im zweiten Kontrollregister wird mit den Bits 2 und 3 die Orientierung der Zeichen verändert, also kann man zum Beispiel schräg schreiben, wie in *Bild 9* dargestellt. Dazu muß vor einem Schreibbefehl der entsprechende Bitcode sedezimal in das unter Adresse 72h erreichbare Register geschrieben werden.
Damit das Programm nicht ständig neu eingetippt werden muß, können wir an die Adresse 8810 an Stelle der Zeicheneingabe einen Unterprogrammaufruf einfügen. In dem Unterprogramm ab der Adresse 8832 (*Bild 10*) wird dann vor der Zeicheneingabe die Schriftart festgelegt. Außerdem wollen wir auch die Schriftgröße verändern, wobei bis zu 16fache Vergrößerung in der X- und Y-Achse, auch unterschiedlich, möglich ist. Dazu benötigen wir das Zeichengrößen-Register (C-Size) mit der Adresse 73h (*Bild 11*). Die Bits 0...3 enthalten den Vergrößerungsfaktor für die Y-Achse, die Bits 4...7 den der X-Achse. Sedezimale Werte von 1...F ergeben bis zu 15fache Vergrößerung in einer Achse, der Wert 00 ergibt die größten Zeichen (16fach).

Befehl Bitfolge				Wert	Funktion
3	2	1	0		a) Vektorbefehle b) Buchstaben
a)					
0	0	0	0	0	durchgezogene Linien
0	0	0	1	1	gepunktete Linien - - - - - -
0	0	1	0	2	gestrichelte Linien — — —
0	0	1	1	3	Kombination ——— — ——— —
b)					
0	0	0	0	0	normale Buchstaben
0	1	0	0	4	schräge Buchstaben
1	0	0	0	8	Buchstaben vertikal
1	1	0	0	C	Buchstaben schräg u. vertikal

Die Bits 4 - 7 sind immer 0

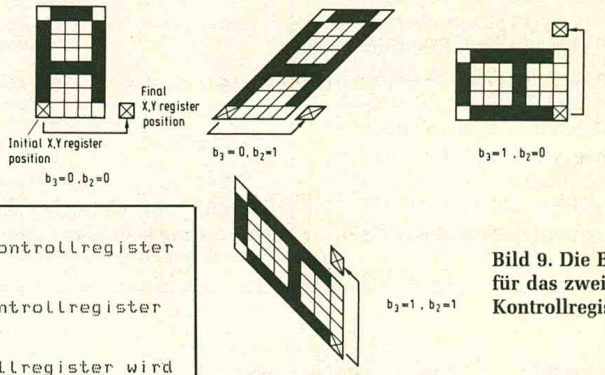


Bild 9. Die Befehle für das zweite Kontrollregister

Bitfolge b3	b2	b1	b0	Wert	Funktion
0	0	0	0	0	Schreibfunktion, im 1.Kontrollregister wird Bit Nr. 1 gesetzt
0	0	0	1	1	Löschfunktion, im 1. Kontrollregister wird Bit Nr. 1 gelöscht
0	0	1	0	2	schreiben, im 1. Kontrollregister wird Bit Nr. 0 gesetzt
0	0	1	1	3	unsichtbar schreiben, im 1. Kontrollregister wird Bit Nr. 0 gelöscht
0	1	0	0	4	Bildschirm löschen
0	1	0	1	5	X- und Y-Register auf 0 setzen
0	1	1	0	6	Bildschirm und X- sowie Y-Register löschen
0	1	1	1	7	Bildschirm und alle Register außer Lichtgriffel löschen, Schriftgröße am kleinsten
1	0	0	0	8	Bildschirm kurz hell für Lichtgriffelposition
1	0	0	1	9	Lichtgriffelfunktion einschalten
1	0	1	0	A	Rechteck ausgeben, Größe P * Q vom Zeichengrößen-Register abhängig
1	0	1	1	B	Quadrat ausgeben, Größe P * Q vom Zeichengrößen-Register abhängig
1	1	0	0	C	Bildschirm hell, Zeichen dunkel
1	1	0	1	D	X-Register löschen
1	1	1	0	E	Y-Register löschen
1	1	1	1	F	Direktzugriff auf Bildschirmspeicher (nur bedingt verwendbar)

Bild 8. Die Befehle für das Befehlsregister (Adresse 70 beim Z80-System)

```
8810    CD 8832     neues Unterprogramm
"        "
8832    3E 04       schräge Zeichen
8834    D3 72       Ausgabe an 2. Kontrollregister
8836    CD 882A     call warte
8839    3E 83       Faktor X*8, Y*3
883B    D3 73       Ausgabe an C-Size
883D    CD 882A     call warte
8840    3E 00       X-Koordinate = 0
8842    D3 79       Ausgabe an X-Register LSB
8844    CD 882A     call warte
8847    3E 00       X-Koordinate = 0
8849    D3 78       Ausgabe an X-Register MSB
884B    CD 882A     call warte
884E    3E 64       Y-Koordinate = 100
8850    D3 7B       Ausgabe an Y-Register LSB
8852    CD 882A     call warte
8855    CD 0024     call Zeicheneingabe
8858    C9          ret
```

Bild 10. Ein Z80-Unterprogramm zur Darstellung verschiedener Zeichenformen an verschiedenen Koordinaten

```
 5  CLS                          :REM Bildschirm löschen
 7  POKE HEX("87C5"),0           :REM Cursor aus
10  GDP = HEX("70")
20  PAGE 0,0                     :REM Bildseite 0
30  OUT GDP+1,11
40  OUT GDP+2,0                  :REM Vektorform
50  MOVETO 200,150               :REM nicht am Rand
                                 anfangen
60  KV = INT((255+1-248)*RND(1)+248) :REM F8-FF
70  FOR I = 1 TO 3               :REM Vektorlänge * 3
80  OUT GDP,KV
85  WAIT GDP,4,0                 :REM kann entfallen
90  NEXT
100 GOTO 60
```

Bild 12. Ein Programm in Basic, das Kurzvektorausgabe benutzt

```
a) Zeichengrößen-Register (73h)

    Bit 7 - 4      Vergrößerungsfaktor P in X-Achsenrichtung

    Bit 3 - 0      Vergrößerungsfaktor Q in Y-Achsenrichtung

Beispiele:

0001 0001   11   kleinste Zeichengröße:  5 * 8 Punktematrix

0000 0000   00   größte Zeichengröße:  16*5 + 16*8   "

1111 1111   FF   Zeichengröße:         15*5 + 15*8   "

0100 1101   4D      "       :          4*5 + 13*8   "
```

```
b) Kurzvektorbefehle für Befehls-Register (70h)

Bit 7                  immer gesetzt = 1

Bit 6-5 = Bit 4-3:  Vektorlänge in Bildpunkten:

0 0                 :  0 Punkte

0 1                 :  1 Bildpunkt

1 0                 :  2 Bildpunkte

1 1                 :  3 Bildpunkte

Bit 2-0 : Richtungsangabe (gleicher Code wie bei
          Vektorbefehle Gruppe C)

Beispiel für eine Vektorlänge von 3 Bildpunkten:

Richtung                  Befehl
                          dez.  sedez.   binär

 3 Uhr                    248   F8       1111 1000

 1 Uhr 30'                249   F9       1111 1001

12 Uhr                    250   FA       1111 1010

10 Uhr 30'                251   FB       1111 1011

 6 Uhr                    252   FC       1111 1100

 4 Uhr 30'                253   FD       1111 1101

 9 Uhr                    254   FE       1111 1110

 7 Uhr 30'                255   FF       1111 1111
```

Bild 11. Das Zeichengrößenregister (a) und die Kurzvektorbefehle (b)

Letztlich soll der Buchstabe an einer bestimmten Bildschirmstelle erscheinen. Dazu verwenden wir das X- und Y-Register. Dieses wird zwar automatisch bei jeder Zeichenausgabe vom Prozessor verändert, kann aber über die Adressen 78h...7Bh (Bild 2) gelesen und korrigiert werden. Die Koordinaten werden in 12 Bits übergeben, damit sind prinzipiell 4096 × 4096 Bildpunkte ansprechbar. Für eine Bildschirmseite mit 256 Bildpunkten (Y-Achse) × 512 Bildpunkten (X-Achse) brauchen wir für die Y-Koordinate nur das niederwertige Byte (LSB). Der Wert FF (sedezimal) ergibt den Bildpunkt in der linken oberen Bildschirmecke. Soll dort der kleinstmögliche Buchstabe stehen, müssen wir natürlich die Buchstabenhöhe mit 8 Bildpunkten von FF abziehen, das ergibt den Wert F7. Nach dem Eingeben des Ergänzungsprogrammes erscheinen trotz Verwendung der X- und Y-Register die Buchstaben noch in der linken unteren Ecke. Warum?

Sehen Sie sich das Unterprogramm „era" an. Der Löschbefehl besitzt den Wert 6 und dabei wird das X- und Y-Register auf 0 gesetzt. An der Adresse 8826 muß daher der Wert 4 stehen, dann erscheinen die Zeichen an der Koordinate X,Y = 0,100. Verwenden wir bei Adresse 8841 den Wert COh, dann erscheinen unsere Zeichen etwa in der Bildschirmmitte. Größere Werte als FF können für die X-Koordinate jetzt nicht verwendet werden. Da benötigen wir das höherwertige Byte (MSB) des X-Registers, das den Wert 1 enthalten muß. Das kleinste Zeichen erscheint dann am rechten Bildrand, wenn wir FAh in das LSB und den Wert 1 in das MSB des X-Registers schreiben. Hierzu muß aber an der Adresse 8833 der Wert 0 und an

883A der sedezimale Wert 11 stehen, sonst reicht das Zeichen über den rechten Bildschirmrand hinaus und ist nur zum Teil sichtbar. Die obere rechte Bildschirmecke besitzt die X,Y-Koordinaten 511,255.

Vektoren über Vektoren

Die besonderen Eigenschaften des Grafik-Prozessors kommen erst bei der Vektordarstellung zum Vorschein. Es gibt wiederum sehr viele Möglichkeiten, wobei wir wieder bei den einfacheren, mit den Kurzvektorbefehlen beginnen wollen. Möglich sind eine Längenangabe (1...3 Bildpunkte), die Strichart, z. B. unterbrochen und acht Richtungen. Bild 11 zeigt auch eine Befehlsübersicht zu den Kurzvektorbefehlen.
Zur Angabe der Kurzvektorbefehle dient wieder das Befehlsregister (CMD) mit der Adresse 70h. Verwendung finden die Bytes 80h...FFh, also mit gesetztem Bit 7. Die Register Delta-X und Delta-Y werden dabei nicht benötigt. Die Bits 0...2 codieren die Richtung, in der gezeichnet wird. Die Bits 3...4 bzw. 5...6 enthalten die Bildpunkteanzahl je Kurzvektor. Die Befehlsausgabe mit den sedezimalen Werten F8...FF ergibt also Vektoren mit einer Länge von drei Bildpunkten in acht verschiedene Richtungen. Die Strichart wird durch das zweite Kontrollregister beeinflußt, wobei die Bits 0 und 1 Verwendung finden.
Nachdem so viel mit der Z80-Maschinensprache gearbeitet wurde, sollen die Kurzvektoren hier mit dem 8-KByte-Basic ausprobiert werden. Mit der RND-Funktion erzeugen wir Zufallszahlen zwischen 248 und 255. Diese Zahlen sind als Kurzvektorbefehle interpretierbar. Die Richtung des Kurzvektors ist also zufällig und die mit dem Programm (*Bild 12*) gezeichnete Bahn entspricht in etwa einer zufälligen Teilchenbewegung in einem Gas.
Bild 13 zeigt, wie es nach kurzer Zeit auf dem Bildschirm aussieht. Damit der Vektor im Bereich der sichtbaren Bildschirmseite bleibt, wird in Zeile 30 das Bit 3 des ersten Kontrollregisters zusätzlich zum Schreibbefehl gesetzt. In Zeile 40 wird die Strichart, hier kontinuierlich, an das zweite Kontrollregister mit Adresse 72h ausgegeben. Ein Programmabbruch erfolgt einfach mit der ESCAPE-Taste.

Lange Vektoren

Nun wird es komplizierter. Es sollen Linien von dem Koordinatenpunkt X1,Y1 zum Punkt X2,Y2 gezeichnet werden. Dazu benötigen wir für die Anfangskoordinaten die Register X und Y, für die Differenz der jeweiligen Achsenkoordinaten (entspricht der skalaren Komponente des Vektors bei senkrechter Projektion) die Register Delta-X und Delta-Y sowie das zweite Kontrollregister für die Strichart, zum Beispiel gepunktete Linien, und dann das Befehlsregister. Die sedezimalen Befehlswerte reichen von 10...1F für diese Vektorbefehle.
Wir verwenden wieder ein Basic-Programm (*Bild 14*), da so die Koordinateneingabe kürzer zu programmieren ist, und geben die Befehle direkt an die Register des Prozessors.

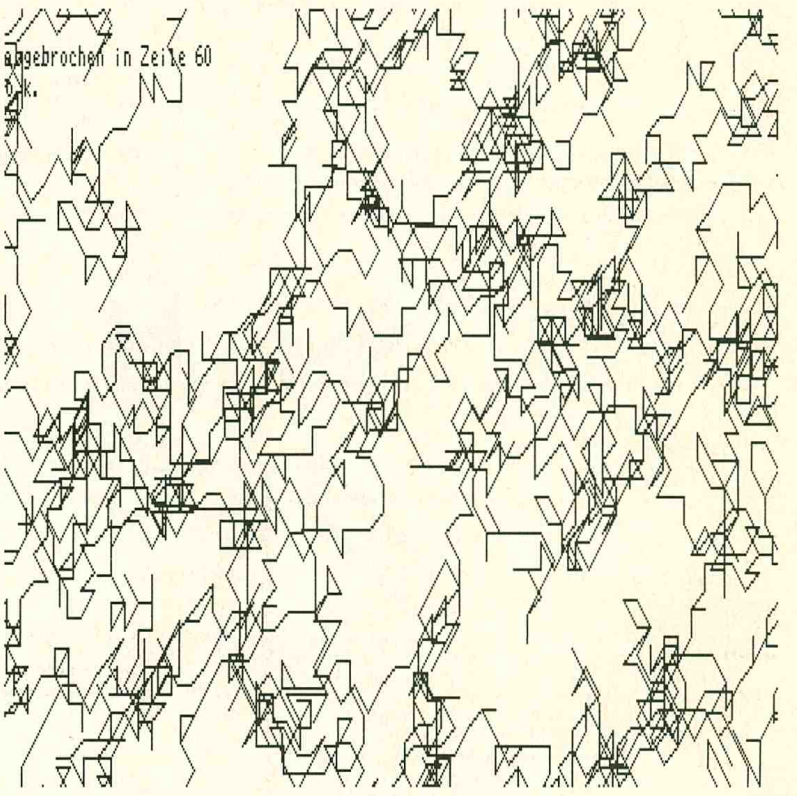


Bild 13. Das zeichnet das Programm aus Bild 12

```
 10 CLRS
 15 POKE HEX("87C5"),0
 20 INPUT"Anfangsadresse X1,Y1";X1,Y1
 30 INPUT"Endadresse     X2,Y2";X2,Y2
 50 GDP = HEX("70")
 60 PAGE 0,0
 70 OUT GDP+1,3           :REM schreiben
 80 OUT GDP+9,X1          :REM X-Register
 90 OUT GDP+11,Y1         :REM Y-Register
100 REM
110 CD = 17               :REM 0001 0111, DX u. DY positiv
120 DX = X2 - X1
130 IF DX<0 THEN CD = 19 :REM 0001 0011, DX negativ
140 ZX = ABS(DX)
150 OUT GDP+5,ZX          :REM Delta X-Register
160 DY = Y2 - Y1
170 IF DY<0 THEN CD = 21 :REM 0001 0101, DY negativ
180 ZY = ABS(DY)
190 OUT GDP+7,ZY          :REM Delta Y-Register
195 REM
200 IF DX<0 AND DY<0 THEN CD = 23 :REM DX u. DY neg.
205 REM
210 WAIT GDP,4,0
220 OUT GDP,CD            :REM Befehlsregister
230 INPUT"NOCHMAL";X$
240 IF X$ = "j" THEN 10
250 END
```

Bild 14. Dieses Programm benutzt Vektorbefehle

Nach Eingabe der Koordinaten in Zeile 20 und 30 wird als erstes das X- und Y-Register mit der Anfangsadresse geladen (Zeile 80 und 90). Falls die Koordinatendifferenz DX bzw. DY negativ wird, muß dies dem Prozessor durch ein gesetztes Bit 1 bzw. Bit 2 im Befehlswort mitgeteilt werden. Daher ergeben sich die vier sedezimalen Befehle 11, 13, 15 und 17, die in *Bild 15*, Gruppe A, enthalten sind. Nach der Absolutwertbildung erfolgt die Ausgabe an die Delta-X- und Delta-Y-Register. Die Zeile 210 kann entfallen, da Basic dem GDP genügend Zeit zur Verarbeitung der Befehle läßt. Mit dem Beispielprogramm kann auf der X-Achse nur der Koordinatenpunkt 255 erreicht werden, da das Delta-Register nur 8 Bit besitzt. Gibt man größere Werte ein, so ergibt sich ein „FC Fehler" (Function call error). Größere Vektoren müssen daher in zwei Teilen gezeichnet werden. *Bild 16* zeigt einen Programmablauf. Mit einer Basic-Zeile, z. B. 75 OUT GDP+2,Wert, kann man entsprechend den Befehlen des zweiten Kontrollregisters die Linienart auswählen.

Die sedezimalen Befehle 10, 12, 14 und 16 (Gruppe B) werden verwendet, wenn das Delta-X-Register oder Delta-Y-Register nicht benötigt werden. Somit ist nur das Vorzeichen des Differenzwertes eines Registers zu berücksichtigen und dementsprechend können die Bits 1 und 2 gesetzt werden.

Noch einfacher sind Vektoren parallel zur Koordinatenachse oder zu Diagonalen mit einer Befehlsgruppe zu zeichnen, die nur das Delta-Register mit der größeren Vektorkomponente auswertet. Entsprechend den acht Richtungsmöglichkeiten gibt es die sedezimalen Befehle 18...1F, die in Bild 15, Gruppe C, gezeigt sind. *Bild 17* zeigt ein Programm zur Ausgabe eines Fadenkreuzes. Die Vektorlänge mit 50 Bildpunkten in der X-Achse wird in das Delta-X-Register geladen. Nach der Berechnung des jeweiligen Startpunktes des Vektors und der Ausgabe in das X- und Y-Register wird der Vektor in die angegebene Richtung gezeichnet.

Der Leser sollte vor allen Dingen den Mut haben, die jeweiligen Parameter der Register in den Beispielprogrammen abzuändern, denn nur durch Experimentieren lernt man den GDP wirklich kennen.

```
Vektor-Befehle: Gruppe A
      Vorzeichen                Befehl:
Delta-X    Delta-Y        dez.   sedez.   binär
   +          +            17     11      0001 0001
   -          +            19     13      0001 0011
   +          -            21     15      0001 0101
   -          -            23     17      0001 0111
-------------------------------------------------------
Vektorbefehle: Gruppe B
Delta-X    Delta-Y        dez.   sedez.   binär
   > 0        -            16     10      0001 0000
   -          > 0          18     12      0001 0010
   -          < 0          20     14      0001 0100
   < 0        -            22     16      0001 0110
-------------------------------------------------------
Vektorbefehle: Gruppe C
Richtung                  Befehl
                          dez.   sedez.   binär
 3 Uhr                    24     18      0001 1000
 1 Uhr 30'                25     19      0001 1001
12 Uhr                    26     1A      0001 1010
10 Uhr 30'                27     1B      0001 1011
 6 Uhr                    28     1C      0001 1100
 4 Uhr 30'                29     1D      0001 1101
 9 Uhr                    30     1E      0001 1110
 7 Uhr 30'                31     1F      0001 1111
```

Bild 15. Eine Zusammenstellung der Vektorbefehle

```
 10 REM Fadenkreuz
 20 CLRS
 30 POKE HEX("87C5"),0
 40 INPUT"Koordinate X,Y";X,Y
 50 GDP = HEX("70")
 60 PAGE 0,0
 70 OUT GDP+1,3                          :REM schreiben
 80 GOSUB 300
 90 GOTO 40
100 REM ************************
300 REM AUSGABE
310 VK = 50                              :REM Strichlänge X-Achse
320 OUT GDP+5,VK                         :REM Delta X-Register
330 KX = X - VK/2: IF KX <0 THEN KX = 0  :REM Startpunkt
340 OUT GDP+9,KX                         :REM X-Register LSB
345 OUT GDP+11,Y                         :REM Y-Register LSB
350 OUT GDP,24                           :REM Vektor nach 3 Uhr
360 WAIT GDP,4,0
390 VK = 25                              :REM Strichlänge Y-Achse
395 OUT GDP+5,VK                         :REM Delta X-Register
400 KY = Y - VK/2:IF KY<0 THEN KY = 0    :REM Startpunkt
410 OUT GDP+9,X                          :REM X-Register LSB
420 OUT GDP+11,KY                        :REM Y-Register LSB
430 OUT GDP,26                           :REM Vektor nach 12 Uhr
440 RETURN
```

Bild 17. Ein Fadenkreuz mit Vektorbefehlen aufgebaut und ausgegeben

```
Anfangsadresse X1,Y1? 50,40
Endadresse     X2,Y2? 200,75
NOCHMAL?
```

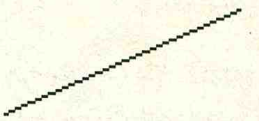


Bild 16. So sieht ein Vektor aus

PAKETANGEBOTE

Die Paketangebote beinhalten nicht nur eine Ersparnis, sondern stellen gleichzeitig Kaufhilfen für den Noch-Nicht-Fachmann dar. Die einzelnen Baugruppen sind sinnvoll zusammengestellt und orientieren sich an der Schulfernseh-Sendereihe.

Auf Wunsch liefern wir die Teile entsprechend dem Fortschritt der Reihe jeweils kurzfristig vor dem Sendetermin. Zur Portoersparnis empfehlen wir eine Einzugsermächtigung. Bitte geben Sie beim Auftrag an, welches Programm Sie verfolgen wollen.

PAKET M — BS: DM 219,90 — FB: DM 333,–

Das kleinste Paket ist auf der hinteren Umschlagseite abgebildet. Es umfaßt den MINIBUS, eine Grundplatte speziell für diese Anordnung, das 5-V-Netzteil, die Z80-CPU mit Speicher auf der SBC 2 und die IOE/EX. Die IOE/EX enthält in festverdrahteter (gedruckter) Form den IOE-Busanschluß und alle nötigen Bauteile für die Experimente bis zur 10. Folge (Roboter). Paket M ist geeignet für alle, die zunächst nicht selbst Programme schreiben, sondern nur den Umgang mit der Hardware üben wollen. Software (EPROMs für die SBC2) muß extra bestellt werden, siehe Preistabelle.

PAKET 1 — BS: DM 849,– — FB: DM 1145,–

Das Z80-Paket geht den Schritt weiter zur eigenen Programmerstellung, enthält also Bildschirmanschluß und Tastatur, das nötige Grundprogramm und das Kassetteninterface. Die IOE hat das übliche Rasterfeld statt der festverdrahteten Experimente.
Paket 1 besteht aus POW5V, SBC2, BUS2I, IOE, KEY, TAST, GDP64K, CAS und MON1 und deckt die ersten 15 Folgen der Sendereihe ab.

PAKET 2 — BS: DM 514,50 — FB: DM 689,–

Erweitert Paket 1 zum Endausbau nach der Fernsehserie. Dazu gehören der Prozessor CPU68K, eine ROA64-Speicherbaugruppe mit dem 68008-Grundprogramm (Grafiktreiber, Editor und Assembler) und einem 8K×8-RAM-Baustein sowie der EPROM-Programmierer.

PAKET 3 — BS: DM 1360,– — FB: DM 1830,–

Alle Artikel aus Paket 1 und 2 zum nochmals vergünstigten Sonderpreis.

PAKET 68 — BS: DM 1950,– — FB: DM 2499,–

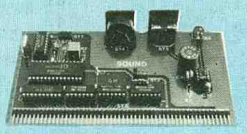
Der große NDR-Klein-Computer mit epromresidentem Assembler und PASCAL/S. Verwendet dynamische RAMs auf der DRAM128, mit gesamt 96 K nutzbarem RAM-Speicher – 32 K sind wegen Überlappung mit dem Grundprogramm ausgeblendet.

Zum Paket gehören:

CPU68K: 68008-CPU
DRAM128: 128 K RAM
ROA64: Grundprogramm
ROA64: PASCAL/S
GDP64K: Grafikinterf.
KEY: Tastaturanschl.
CAS: Kassetteninterf.
POW5V: Netzteil
TAST: Tastatur mit Gehäuse und Kabel
BUS2III: 12er-BUS

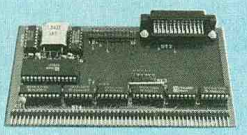
Das Paket 68 eignet sich besonders für mikrocomputererfahrene Anwender, die den 68000-Prozessor kennenlernen wollen, oder für hauptsächlich an Software-PASCAL, Grafik-Interessierte.

SOUND

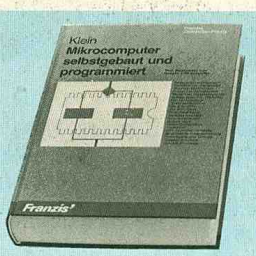


Die Schaltung enthält neben drei in Tonhöhe und Lautstärke programmierbaren Tongeneratoren einen in Grundfrequenz und Amplitude einstellbaren Rauschgenerator. Diese Signalquellen lassen sich einzeln oder gemischt auf einen NF-Ausgang schalten. Alternativ zur programmgesteuerten Lautstärkeeinstellung besteht die Möglichkeit, die Amplitude der Tonfrequenz mit einer von zehn fest eingespeicherten Hüllkurven zu modulieren.

SER



Serielle Datenübertragung wird zur Kommunikation mit Peripheriegeräten eingesetzt, dabei vorzugsweise zur Überbrückung größerer Entfernungen. Der auf SER eingesetzte Serienwandler (UART) ermöglicht asynchronen Betrieb und ist per Programm auf verschiedene Übertragungsgeschwindigkeiten (Baudraten) und Rahmenformate einstellbar. RS232/V28-Leitungstreiber sorgen für die elektrische Anpassung an eine weitverbreitete Norm.



Bausätze zum Bu Schulfe

GDP 64 K

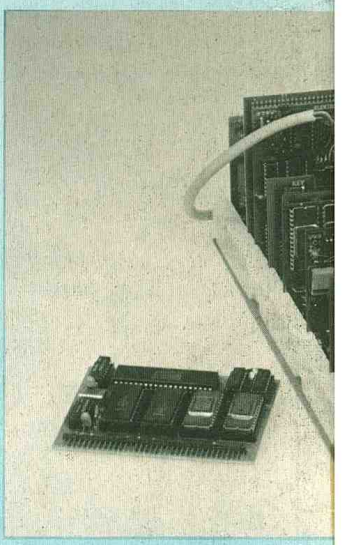
Bildschirm-Steuereinheit für hochauflösende Grafik und Textanzeige. Als Bindeglied zwischen Computer und Bildschirm-Monitor zusammen mit der Tastatur Voraussetzung für die Eingabe von eigenen Programmen oder Daten. Die GDP64K besitzt Speicher für vier Bildschirmseiten (64 KByte), um schnell bewegte Grafik darstellen zu können; ist allerdings auch die teuerste Baugruppe im Paket.

ROA 64

Einfache und billige Speicherkarte für gemischte Bestückung mit EPROMs und RAMs. Dient beispielsweise der Aufnahme des 68008-Grundprogramms mit Editor und Assembler sowie der zugehörigen RAMs. Gesamtkapazität 64 KByte, die innerhalb des 1-MByte-Adreßraums frei angeordnet werden kann. Weitere ROA64s werden z. B. für die Aufnahme der PASCAL/S-EPROMs benötigt.

SBC 2

Die einzige Baugruppe, die beim Umstieg auf den 16-Bit-Prozessor überflüssig wird: Die Z80-CPU. Neben dem 8-Bit-Prozessor sind 4 K RAM und Sockel für 8 K EPROM auf der SBC2. Mit dem Z80 laufen u. a. die Versuche Musik und Ampelsteuerung, der Roboter und das Monitorprogramm für Programmierung in Maschinensprache.



PAKET 3

Das Kombipaket (1 + 2), wie oben abgebildet, umfaßt alle in der Sendereihe benutzten Baugruppen bis auf den 21-/26-V-Wandler für den EPROM-Pro-

Preisliste Baugruppen

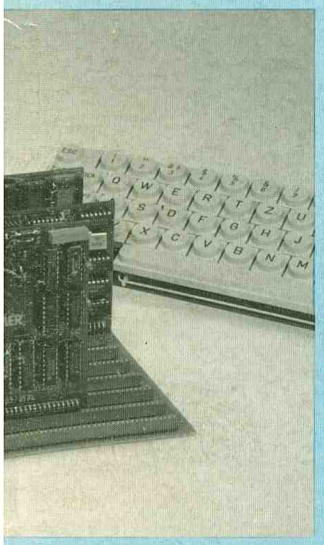
Nach dem Stand vom 15. Januar 1985

Typ	Funktion	P	Einzelpreise inkl. MwSt. LP	BS	FB
POW5V	Spannungsversorgung 5 V/3 A	M,1,68	15,–	39,95	58,50
SBC2	Z80A-CPU, 4 K RAM, EPROM-Sockel	M,1	15,–	79,95	129,–
BUS2I	Grundpl. 6 Plätze, 4 teilbest.	1	15,–	39,95	58,50
BUS2II	Grundpl. 6 Plätze, vollbestückt	–	15,–	68,90	89,–
BUS2III	Grundpl. 12 Plätze, vollbestückt	68	30,–	137,80	169,–
BUS2IV	Grundpl. 18 Plätze, vollbestückt	–	45,–	206,70	249,–
MINIBUS	Grundpl. für POW5V, SBC2, IOE	M	–	19,50	24,50
IOE	16-Bit-Ein-, 16-Bit-Ausgabe	1	15,–	39,95	69,–
IOE-EX	IOE f. Exp. Musik, Ampel, Robot.	M	15,–	89,90	145,–
GDP64K	Vollgrafik m. 64-KByte-Bildsp.	1,68	15,–	359,–	458,–
KEY	Tastaturanschluß	1,68	15,–	49,95	89,–
TAST	DIN-Tastatur o. Funktionstasten	1,68	–	–	198,–
TAST/G	Gehäuse zur kleinen Tastatur	68	–	–	49,90
DINTAST	Große Tast. m. Funktionstasten	–	–	–	410,40
GEH/DT	Gehäuse zur großen Tastatur	–	–	–	112,86
TAST/K	Rundkabel für Anschluß an KEY	68	–	–	12,50
CAS	Kassettenrecorder-Anschluß	1,68	15,–	74,90	129,–
CPU68K	16-Bit-Prozessor 68008	2,68	15,–	199,–	265,–
ROA64	Speicherkarte für 8 K × 8 RAM/EPROM	2,68	15,–	39,95	84,–
DRAM64	128 K dyn. RAM-Speicher, 64-K-bestückt	–	15,–	398,–	479,–
DRAM128	128 K dyn. RAM-Speicher, vollbestückt	68	15,–	599,–	699,–
PROMER	EPROM-Programmierzusatz	2	15,–	79,95	129,–
POW21/26	Spannungswandler für PROMER	–	15,–	58,60	76,–
BANKBOOT	A16-A19-Erw. f. Z80, Floppy-Booter	–	15,–	64,50	94,–
CPUZ80	Z80 Vollausbau	–	15,–	74,50	129,–
FDC	Floppy-Steuerbaugruppe 3,5"/5,25"/8"	–	15,–	349,–	439,–
AD8 × 16	16fach-Anal.-/Dig.-Wandler 8 Bit	–	15,–	148,–	215,–
AD10 × 1	Analog-/Digitalwandler 10 Bit	–	15,–	265,–	384,–
DA	2 Digital-/Analogwandler 8 Bit	–	15,–	139,50	208,–
SER	Serielle Schnittstelle 6551	–	15,–	98,50	145,–
SOUND	AY-3-8912, Soundgenerator	–	15,–	89,90	129,–
CENT	Aufsteckplatine für IOE	–	–	10,–	20,–

P = Paketzugehörigkeit, LP = Leiterplatte, BS = Bausatz, FB = Fertigbaugruppe

ELEKTRONIKLADEN 4930 DETMOLD 18 ☎ 05232/8171
Verkaufsstelle München: Schulstr. 28, 8000 München 19, Tel. 0 89/1 67 94 99

d Geräte und zur ehreihe



grammierer. Anders als auf dem Foto wird allerdings zum Paketpreis nur eine Grundplatte mit sechs Steckplätzen (BUS2II) geliefert. Wir empfehlen gegen Aufpreis den 12er-Bus (BUS2III).

III
Mikroelektronik

PROMER

Bei Spannungsausfall sind alle Programme im RAM gelöscht. EPROMs halten dagegen den Inhalt, bis sie mit reichlich UV-Licht bestrahlt werden. EPROMs. können nur in einem speziellen Programmierverfahren beschrieben werden, wozu diese Baugruppe dient. Programmierbar sind 2716, 2732 und 2764.

CPU 68 K

Die „große" Prozessorkarte benutzt den 68008, eine Sonderversion der bekannten 68000-CPU mit 8-Bit-Datenbus. Jeder 16-Bit-Zugriff wird in zwei 8-Bit-Hälften aufgeteilt, die kurz nacheinander ausgeführt werden. Neben Geschwindigkeit sind der große Adreßbereich und klare Befehlsstruktur die wesentlichsten Vorteile des 68008 gegenüber dem Z80.

CAS

Ohne externe Speichermöglichkeit wären alle ins RAM eingegebenen Programme nach dem Stromausschalten verschwunden. Mit CAS ist es möglich, Programme oder Daten mit einem Kassettenrecorder abzuspeichern. Die Computerdaten werden in eine Serie von Einzelbits zerlegt und auf einen hörbaren Ton moduliert.

Software und Begleitmaterial

Typ	Funktion	Einzelpreise inkl. MwSt. P	LP	BS	FB
MON1	Monitorprogramm für SBC2	1	–	–	60,–
MUO	Musik-Testprogr. SBC2 o. RAM	–	–	–	30,–
MUM	Musik-Testprogr. SBC2 m. RAM	–	–	–	30,–
AMPEL	Programm für Experiment Ampel	–	–	–	30,–
ROBOT	Programm für Experiment Roboter	–	–	–	30,–
BASIC	BASIC für SBC2, mit Handbuch	–	–	–	75,–
GOSI	Grafiksprache f. SBC2, m. Handb.	–	–	–	75,–
MON68K	68008-Monitor/-Editor/-Assembler	2,68	–	–	155,–
PASCAL	68008-PASCAL/S	68	–	–	155,–
8K-RAM	8 K × 8 stat. RAM	2	–	–	70,–
DIS4.1	Disassembler	–	–	–	55,–

Begleitmaterial

BUCH	R.-D. Klein: Mikrocomputer selbstgebaut u. programm.	38,–
VCAS	2 Videokassetten mit der Sendereihe (VHS, Beta, V2000)	248,–
MC-SCH	mc-Begleitheft Schaltpläne und Unterlagen	8,–
MC-Z80G	mc-Begleitheft Z80-Grundprogramme	12,–
MC-Z80A	mc-Begleitheft Z80-Aufbauprogramme	12,–
MC-68G	mc-Begleitheft 68008-Grundprogramme	12,–
MC-68A	mc-Begleitheft 68008-Aufbauprogramme	12,–
MC-PAS	mc-Begleitheft PASCAL/S-Quellprogrammliste	12,–
VALVO-Buch	16-Bit-Mikroprozessor „Befehlsvorrat"	24,–
VALVO-Buch	16-Bit-Mikroprozessor „Eigenschaften"	13,–

Zubehör

ZVM123	Bildschirm-Monitor 12", grün	293,–
ZEUG	Werkzeugsatz mit Lötkolben, Zangen, Draht usw.	69,95
BU18	Buchsenleiste 18polig für BUS2	3,80
ROBOT	Fischertechnik-Roboterbausatz	194,50
EXMUS	Bauteilsatz Experiment Musik, inkl. Eprom MUO o. MUM	41,20
EXAMP	Bauteilsatz Experiment Ampel, inkl. Eprom	34,50
EXROB	Bauteilsatz Experiment Roboter, inkl. Eprom	64,80
NETZ	Großes Schaltnetzteil 5 V/6 A, 12 V/3 A, –5 u. –12 V/0, 5 A	198,–

Alle Preise einschließlich Mehrwertsteuer.

Kostenlose Broschüre bitte anfordern!

Der NDR-Klein Computer

Die Schulfernsehserie Mikroelektronik bietet die einmalige Gelegenheit, Mikrocomputer im Detail verstehen zu lernen, mit aller denkbaren Unterstützung:

a) Fernsehsendung in 26 Folgen
b) Videokassetten davon
c) Buch von R. D. Klein
d) Begleitung durch mc und ELO
e) mc-Sonderhefte und nicht zuletzt:
f) Hardware und Anleitungen vom ELEKTRONIKLADEN

Die Sendereihe ist nicht, wie häufig, auf die Bedienung von Computern abgestellt, sondern auf das wirkliche Verstehen auch der zugrundeliegenden Elektronik. Dazu wird vom Zuschauer erwartet, daß er sein Lerninstrument von Anfang an selbst zusammenlötet und testet. Damit eignet sich der NDR-Klein-Computer nicht nur für die primären Adressaten, die Schüler weiterbildender und berufsbildender Schulen, sondern in besonderem Maße auch für Lehrwerkstätten und branchenfremde Ingenieure, die den Mikroelektronik-Zug nicht verpassen wollen.

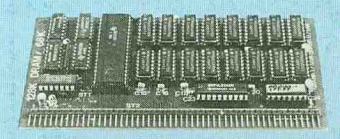
Die „Hardware"

Der Computer wird aufgebaut in vielen übeschaubaren Baugruppen, die auch der logischen Aufteilung zwischen Prozessor, Speicher und Ein-/Ausgabe (E/A) entsprechen.

Durch diese Aufteilung in Einzelbaugruppen kann der Aufbau des Computers Schritt für Schritt vorgenommen werden, sowie die Kenntnisse fortschreiten. Gleichzeitig ergibt sich dadurch große Flexibilität hinsichtlich Um- und Ausbau.

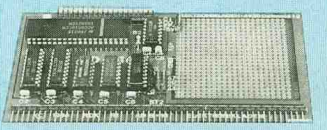
So kann der Computer auch weit über den Lehreffekt hinaus sinnvoller Verwendung zugeführt werden, wobei der finanzielle Mehraufwand für die Lernphase, z. B. bei einem bis zum CP/m-Floppy-Computer ausgebauten Gerät, schließlich vernachlässigbar wird.

BESTELLKARTE



DRAM

Statische RAMs sind etwa vierfach teurer als die aufwendig anzusteuernden dynamischen RAMs. Bei großem Speicherbedarf lohnt sich aber der Aufwand. DRAM128 hat eine Kapazität von 128KByte und kann überlappend mit EPROMs auf ROA-Karten oder mit der Bankboot-Karte arbeiten.



AD 8 × 16

Um Temperaturen, Spannungen und andere Größen zu messen, die nicht nur Ein-/Aus-Zustände annehmen, muß ein Analog-/Digital-Wandler eingesetzt werden. AD 8 × 16 hat 16 Eingänge und unterteilt die Eingangsspannung (Bereich 0–5 V) in 256 Schritte (8 Bit). Auf dem Rasterfeld wird die Elektronik zur Umsetzung der Eingangssignale auf 0–5 V aufgebaut.

DIN-TAST

Für Vielschreiber und kommerzielle Anwendungen empfiehlt sich die große Tastatur mit Funktionstasten, Cursorfeld und Nummernfeld. Diese Sondertasten sind benutzerprogrammierbar (EPROM) und für vier Anwendungen (z. B. PASCAL, Textverarbeitung, Assembler...) umschaltbar. Ein CP/M-Programm unterstützt die Programmierung des EPROMs. Eine Schnellspeicher-Betriebsart kann 20 Zeichen zu jeder Zeit änderbar aufnehmen. Bitte Sonderprospekt anfordern.

ELEKTRONIKLADEN 4930 DETMOLD 18 ☎ 05232/81 71
Verkaufsstelle München: Schulstr. 28, 8000 München 19, Tel. 0 89/1 67 94 99

Wenn Sie wirklich wissen wollen, wie ein Computer funktioniert – bauen Sie ihn doch einfach selbst!

Der NDR-Computer

vom günstigen Einsteigerpaket,
auch für Anfänger, bis zum
professionellen System für Profis.

Die Bausätze und Fertiggeräte erhalten Sie bei uns in gewohnter Spitzenqualität aus erster Hand.

Mehr Information erhalten Sie nach Einsenden anhängender Postkarte.

Graf Elektronik Systeme GmbH, Postfach 1610, 8960 Kempten, Tel. (08 31) 62 11

Filiale Hamburg, Ehrenbergstr. 56, 2000 Hamburg 50 (Altona), Telefon (0 40) 38 81 51
Filiale München, Georgenstr. 61, 8000 München 40 (Schwabing), Telefon (0 89) 2 71 58 58

2800 Bremen, GMCP GmbH, Werftstr. 160, 2800 Bremen 21, Tel. (04 21) 61 30 15
A-1030 Wien, ISYS-Informationssysteme, Landstraßer Hauptstraße 2a, Tel. 75 33 25 CH-4106 Therwil, SYSTECH, Postfach, Tel. (0 61) 73 49 73

ges